Contents

[1. INTRODUCTION 3](#_Toc482381180)

[2. AIMS AND OBJECTIVES 4](#_Toc482381181)

[3. LITERATURE SURVEY 5](#_Toc482381182)

[3.1 DIFFERENT ATTACKS: 6](#_Toc482381183)

[3.1.1 DICTIONARY ATTACK 6](#_Toc482381184)

[3.1.2 BRUTE FORCE ATTACK 6](#_Toc482381185)

[3.1.3 SPYWARE ATTACK 7](#_Toc482381186)

[3.1.4 SHOULDER SURFING 7](#_Toc482381187)

[3.2 EXISTING SYSTEM 7](#_Toc482381188)

[4 PROBLEM STATEMENT 9](#_Toc482381189)

[5 SCOPE OF THE SYSTEM 10](#_Toc482381190)

[6 PROPOSED SYSTEM 11](#_Toc482381191)

[7 ANALYSIS & DESIGN DETAILS 13](#_Toc482381192)

[7.1 Physical Design 13](#_Toc482381193)

[7.1.1 Waterfall Model 13](#_Toc482381194)

[7.2 Logical Design 15](#_Toc482381195)

[7.2.1 Activity Diagram 16](#_Toc482381196)

[7.2.2 Use Case Diagram 17](#_Toc482381197)

[7.2.3 Sequence Diagram 19](#_Toc482381198)

[7.2.4 State Diagram 21](#_Toc482381199)

[7.2.5 Component Diagram 22](#_Toc482381200)

[8 Technologies Used 23](#_Toc482381201)

[8.1 HTML 23](#_Toc482381202)

[8.2 CSS 23](#_Toc482381203)

[8.3 JavaScript/Jquery 23](#_Toc482381204)

[8.4 NodeJs 23](#_Toc482381205)

[8.5 MongoDb 23](#_Toc482381206)

[9 METHODOLOGY: 24](#_Toc482381207)

[9.1 HOVERING 24](#_Toc482381208)

[9.2 Random 26](#_Toc482381209)

[10 Hybrid Textual Authentication Scheme 28](#_Toc482381210)

[11 Resilience of proposed system against attacks 30](#_Toc482381211)

[12 IMPLEMENTATION DETAILS 31](#_Toc482381212)

[12.1 Workflow Diagram 31](#_Toc482381213)

[13 Screenshot 32](#_Toc482381214)

[13.1 Home Page 32](#_Toc482381215)

[13.2 Enter username for Signup. 32](#_Toc482381216)

[13.3 Select any four colors from the grid 33](#_Toc482381217)

[13.4 After selecting four colors 33](#_Toc482381218)

[13.5 Assign numbers to that colors (insert only single value) 34](#_Toc482381219)

[13.6 Save and signup successful 34](#_Toc482381220)

[13.7 Draw pattern on the grid of selected colors 35](#_Toc482381221)

[13.8 Draw valid pattern of selected colors 35](#_Toc482381222)

[13.9 Enter Pin for selected colors. 36](#_Toc482381223)

[13.10 Login successful 36](#_Toc482381224)

[14 TimeLine Chart 37](#_Toc482381225)

[15 Project time line and task distribution 38](#_Toc482381226)

[16 Software testing 39](#_Toc482381227)

[16.1 Stress Testing 39](#_Toc482381228)

[16.2 User Acceptance testing (UAT ) 39](#_Toc482381229)

[16.3 Vulnerability Testing 39](#_Toc482381230)

[16.4 System Testing 40](#_Toc482381231)

[16.5 Smoke testing 40](#_Toc482381232)

[16.6 Security Testing 40](#_Toc482381233)

[16.7 Acceptance Testing 40](#_Toc482381234)

[16.8 Beta Testing 41](#_Toc482381235)

[16.9 Black Box testing 41](#_Toc482381236)

[16.10 GUI (Graphical User Interface) testing 41](#_Toc482381237)

[16.11 Glass box Testing 41](#_Toc482381238)

[16.12 Integration Testing 41](#_Toc482381239)

[16.13 TEST CASES. 42](#_Toc482381240)

[17 Conclusion 44](#_Toc482381241)

[18 REFERENCE: 45](#_Toc482381242)

# INTRODUCTION

Passwords are used to provide authentication in any system like a mobile device. Alphanumeric passwords are in use for user authentication. While today other methods including biometrics and smart cards are possible alternatives, passwords are likely to remain dominant for some time because of concerns about reliability, privacy, security, and ease of use of other technologies. However, in the use of passwords dilemmas often arise in the tradeoff between security and usability. The dilemma arises because passwords are expected to comply with two basic conflicting requirements:

**(1)** Passwords must be easy to recall and remember and

**(2)** Passwords should be secure, i.e., they should look random and should be hard to guess.

They should be changed frequently, and should be different on different accounts of the same user; they should not be written down or stored in plain text. Because it is difficult for humans to remember random strings, users tend to ignore requirements for secure passwords. This leads to poor password practices, including short, simple passwords that are easy to break either by a dictionary attack or personal knowledge of the password owner, use of the same password over months or years, reuse of identical or nearly identical passwords on multiple systems, and propensity to write down passwords and store them insecurely, e.g., a text file containing the users passwords stored on insecure computers or PDAs, post its notes stuck on or near the computer monitor or inside a desk drawer. In an effort to improve password security by making passwords easier to remember, graphical passwords have been proposed. In a typical graphical password scheme, a user chooses several colors present in a grid or a matrix to be his or her password. When logging in, the user must click on the password colors among a larger group of distractor colors. If the user chooses the correct colors, he or she is authenticated. User’s memory for a graphical password may be better than for an alphanumeric password. Secure alphanumeric passwords (i.e., random strings) are based on ability to recall from memory, a task that is difficult for humans. By contrast, graphical passwords are based on recognition of previously known images, a skill at which humans are proficient. However, the problem of shoulder- surfing is a recognized drawback of graphical passwords.

# AIMS AND OBJECTIVES

1. The graphical passwords provide better security against dictionary and brute force attacks as login grid changes for every session.
2. It is very economic and very useful to user because it does not use any sophisticated devices or technology for implementation thus people implementing it can use the system even if they have minimal computer knowledge and would not have to go through any kind of training to use the system.
3. Compare two systems hybrid textual authentication scheme and color grid pattern matching.
4. Passwords must be easy to recall and remember and Passwords should be secure, i.e., they should look random and should be hard to guess.
5. The aim of the project is to implement two types of graphical password authentication techniques i.e. color grid based and pattern based.
6. The system also aims at integrating the two techniques and comparing all the schemes to determine which is a better one.

# LITERATURE SURVEY

Reference [1] describes that an authentication is a process by which system verifies the identity of a user. Authentication is the main step of any security system. The idea of graphical passwords was initially developed by Greg Blonder in the year 1996. Though Graphical passwords are much easier to remember, they provide high level of security. Graphical passwords are classified into two types i.e. recognition based graphical passwords and recall based graphical passwords. Recognition-based systems, also known as cognometric systems or search metric systems. This 2 steps random colored grid graphical password authentication scheme shows promise as a usable and memorable authentication mechanism. Much more research and user studies are needed for graphical password mechanisms to achieve higher level of maturity and usefulness.

Reference [2] describes that using recognition-based techniques, a user is presented with a number of images and the user passes the authentication by recognizing and identifying the images he or she selected during the registration stage. During the password creation, the user has to select the theme first which consists of thumbnail photos. The user then selects and registers a sequence of the selected thumbnail photo to form a password. The user needs to recognize and identify the previously seen photos and touch it in the correct sequence using a stylus in order to be authenticated. Basically, Pass faces works as follows; users are required to select the previously seen human face from a grid of nine faces one of which is known while the rest are decoys. This step is continuously repeated until all the four faces are identified. Results also showed that Pass faces took a much longer login time than textual passwords.

Reference [3] describes that graphical passwords have gained much attention as potential alternatives to text-based passwords. There are three categories of graphical passwords; those using either recognition-based, cued-recall based or recall based techniques. In recognition based systems, a user is presented with a set of images and the user passes the authentication by recognizing and identifying the images he/she had previously selected. In cued-recall systems, users are required to remember and target specific locations within an image. In recall-based systems, however, a user is asked to reproduce something that he/she created or selected earlier during the registration stage. The paper has presented three new shoulder surfing defense techniques designed for recall-based graphical passwords, as well as two experimental evaluations of these techniques. These are Line Snaking defense technique, the Disappearing Stroke technique and the Decoy Stroke technique.

Reference [4] describes that many different types of graphical passwords have been proposed to date; among the more popular approaches in the literature is Pass Points. Pass Points and other click-based graphical password schemes require users to click on a sequence of points on one or more background images. Pass Points usability studies have been performed to determine the optimal amount of error tolerance based on click-point accuracy login and creation times, login error rates, memorability, and general perception. The most effective attacks against click-based (or cued-recall) graphical passwords have been examined.

## **DIFFERENT ATTACKS:**

### **DICTIONARY ATTACK**

A dictionary attack is a method of breaking into a password-protected computer or server by systematically entering every word in a dictionary as a [password](http://searchsecurity.techtarget.com/definition/password). Dictionary attacks work because many computer users and businesses insist on using ordinary words as passwords. Dictionary attacks are rarely successful against systems that employ multiple-word phrases, and unsuccessful against systems that employ random combinations of uppercase and lowercase letters mixed up with numerals.

### **BRUTE FORCE ATTACK**

A simple brute-force attack may have a dictionary of all words or commonly used passwords and cycle through those words until it gains access to the account. A more complex brute-force attack involves trying every key combination in an effort to find the correct password that will unlock the encryption. Due to number of possible combinations of letters, numbers, and symbols, a brute force attack can take a long time to complete.

### **SPYWARE ATTACK**

A spyware is software that, from a user’s perspective, covertly gathers information about a computer’s use and relays that information back to a third party. Spyware has gradually become one of the most common security threats to computer systems. Password collection by spywares has rapidly increased. The research community has expended much effort on this topic. However, how to protect passwords effectively against spyware attack continues to be a problem.

### **SHOULDER SURFING**

Shoulder surfing is using direct observation techniques, such as looking over someone's shoulder, to get information. Shoulder surfing is an effective way to get information in crowded places because it's relatively easy to stand next to someone and watch as they fill out a form, enter a PIN number at an ATM machine, or use a calling card at a public pay phone. Shoulder surfing can also be done long distance with the aid of binoculars or other vision-enhancing devices. To prevent shoulder surfing, experts recommend that you shield paperwork or your keypad from view by using your body or cupping your hand.

## **EXISTING SYSTEM**

1. The existing system will check that the user is already registered or not.
2. If yes, then it will go to the login step, but if user is not registered then user will go to the registration phase.
3. Then at the time of transaction the color pairs and grid will be shown and from that the user will enter the session password. This password will be verified at the verification phase. If the user wants another session then a new session will be generated and the grid and color pair will be shown to the user again.

START

User Registered or no?

Rating of colors

NO

Registration

YES

Login

Grid & color pair are shown

Select color and its number

Verification

Correct?

NO

YES

Stop

**Figure 1: Flowchart of existing system**

# PROBLEM STATEMENT

Passwords are the most commonly used method for identifying users in computer and communication systems. For authentication mostly textual passwords are used. Such passwords have the disadvantage of being hard to remember. Also, alphanumeric passwords are prone to various attacks like dictionary attacks, brute force attacks, phishing, shoulder surfing, etc.[2]

The main problem with the alphanumeric passwords is that once a password has been chosen and learned the user must be able to recall it to log in. But, people regularly forget their passwords. If a password is not frequently used it will be even more susceptible to forgetting. [4]

User authentication is a most important component in most computer security. It provides user with access control and user liability. To make a user more secure and authentic while being online, an alternative to textual passwords must be proposed.[2]

The project aims at implementing color grid based password authentication instead of the conventional alphanumeric ones. Graphical passwords are harder to guess or broken by brute force. The basic need for graphical password is that graphical passwords are expected to be easier to recall, less likely to be written down and have the potential to provide a richer symbol space than text based password. [4]

Users will be hugely benefitted from this project as they won’t have to remember complex textual passwords. Also, users will be more secure using color grid password authentication. So the project contains two steps for authentication: first making the pattern and then assigning the value to color. This approach increases the security.

The project also aims at comparing these two methods i.e. color grid and pattern matching and determining which method would be better.

# SCOPE OF THE SYSTEM

The project is based on color grid password authentication scheme which is an integration of two password authentication schemes i.e. Hybrid textual password authentication scheme and pattern based password authentication scheme. The project will integrate the two above mentioned password schemes. Both the authentication schemes were vulnerable to many attacks which can be overcome by integrating both of them and using them on a single standalone application.

# PROPOSED SYSTEM

As organizations look to improve the security of their operations, they are turning to Technology for a solution. Using graphical passwords to maintain security and make their system robust.

Integrating two password authentication scheme, hybrid textual authentication scheme and pattern matching, this will make the authentication highly secured. Considering the drawbacks of the existing graphical password systems, this project proposed a robust

Graphical password scheme, which is highly adaptable for traditional desktop systems, consists of following steps:

1. The user has to register first.
   1. He/she has to enter a username in a textbox provided by the system.
   2. For password selection, there will be a 3\*3 color grid and the user has to select any four colors of his/her choice.
   3. The selected colors will then appear in a 1\*4 grid.
   4. The user will have to rate the four colors i.e. assign numbers to the colors in the four textboxes provided below the colors.
   5. All this information will be stored in the database of the system.
2. Then the user has to sign in.

2.1) The user has to first enter his/her username in the textbox.

2.2) Then the user has to select the four colors which he/she had chosen previously from a 3\*3 color grid.

2.3) The user has to select the colors in the sequence he had first selected else the system will show an error.

2.4) If the sequence is proper, the user has to enter the numbers associated with the respective color.

2.5) If the numbers entered are correct, the user is verified and signing in is successful; else error is shown.

.

START

NO

User Registered?

SIGNUP

Select Colors from Grid

YES

LOGIN

ENTER USERNAME

Rate the colors from grid

Select the colors from random grid

Check the sequence of colors?

YES

Colors get displayed on a grid

Rate the colors

NO

Validation?

YES

END

User logged in

**Figure 2 : Flowchart of proposed system**

# ANALYSIS & DESIGN DETAILS

Design is the first step into the development phase for any engineered product or system.

Design is a creative process. A good design is the key to effective system. The term

“design” is defined as “the process of applying various techniques and principles for the

purpose of defining a process or a system in sufficient detail to permit its physical

realization”. It may be defined as a process of applying various techniques and principles

for the purpose of defining a device, a process or a system in sufficient detail to permit its

physical realization. Software design sits at the technical kernel of the software

engineering process and is applied regardless of the development paradigm that is used.

The system design develops the architectural detail required to build a system or product.

As in the case of any systematic approach, this software too has undergone the best

possible design phase fine tuning all efficiency, performance and accuracy levels. The

design phase is a transition from a user oriented document to a document to the

programmers or database personnel. System design goes through two phases of

development: Logical and Physical Design.

## **Physical Design**

### **Waterfall Model**

**Analysis Phase**

**Design Phase**

**Coding Phase**

**Testing Phase**

**Figure 3. Waterfall Model**

#### **Analysis Phase:**

To attack a problem by breaking it into sub-problems. The objective of analysis is to determine exactly what must be done to solve the problem. Typically, the system’s logical elements (its boundaries, processes, and data) are defined during analysis.

#### **Design Phase:**

The objective of design is to determine how the problem will be solved. During design the analyst’s focus shifts from the logical to the physical. Data elements are grouped to form physical data structures, screens, reports, files and databases.

#### **Coding Phase:**

The system is created during this phase. Programs are coded, debugged, documented, and tested. New hardware is selected and ordered. Procedures are written and tested. End-user documentation is prepared. Databases and files are initialized. Users are trained.

#### **Testing Phase:**

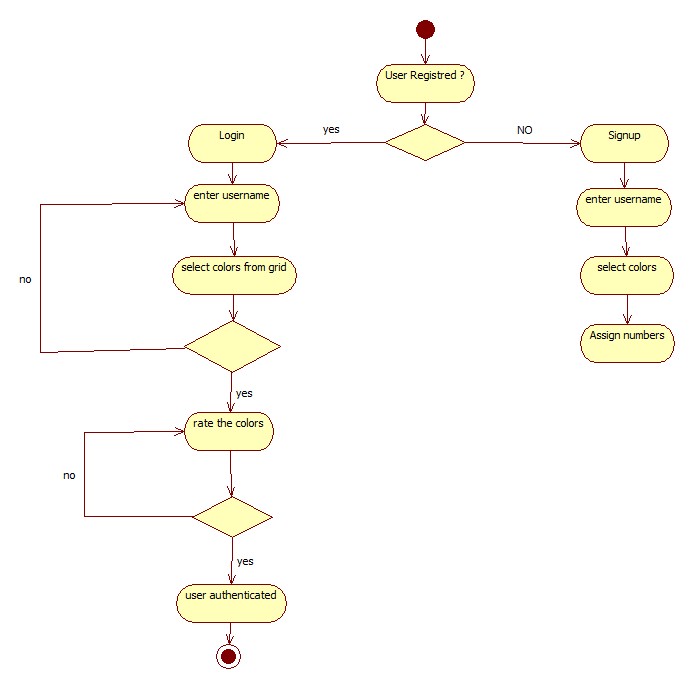
Once the system is developed, it is tested to ensure that it does what it was designed to do. After the system passes its final test and any remaining problems are corrected, the system is implemented and released to the user.

## **Logical Design**

The logical flow of a system and define the boundaries of a system. It includes the following steps:

* Reviews the current physical system – its data flows, file content, volumes, frequencies etc.
* Prepares output specifications – that is, determines the format, content and frequency of reports.
* Prepares input specifications – format, content and most of the input functions.
* Prepares edit, security and control specifications.
* Specifies the implementation plan.
* Prepares a logical design walk through of the information flow, output, input, controls and implementation plan. Reviews benefits, costs, target dates and system constraint

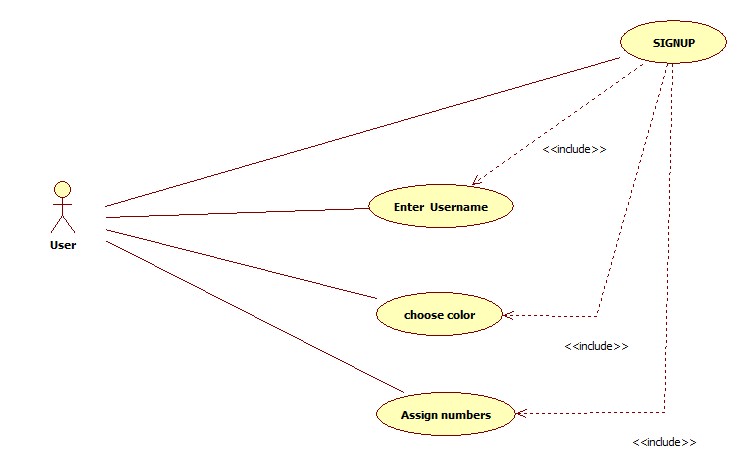
### **Activity Diagram**

An activity diagram is essentially a flowchart, showing flow of control from activity to activity. This involves modeling the sequential (and possibly concurrent) steps in a computational process. With an activity diagram, you can also model the flow of an object as it moves from state to state at different points in the flow of control

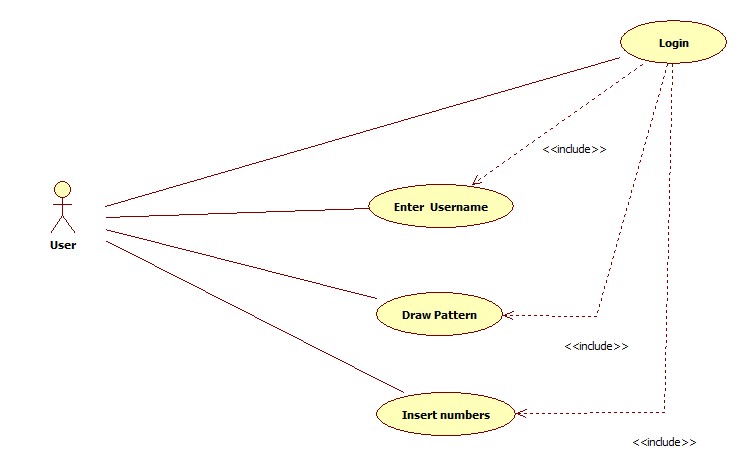
**Figure 4. Activity Diagram**

### **Use Case Diagram**

Use case diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of systems. Use case diagrams are central to modeling the behavior of a system, a subsystem, or a class. Each one shows a set of use cases, actors and their relationships.



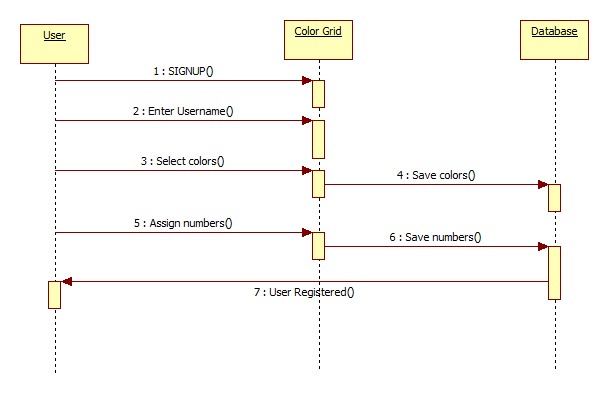
**Figure 5. Use Case Diagram for Signup**

****

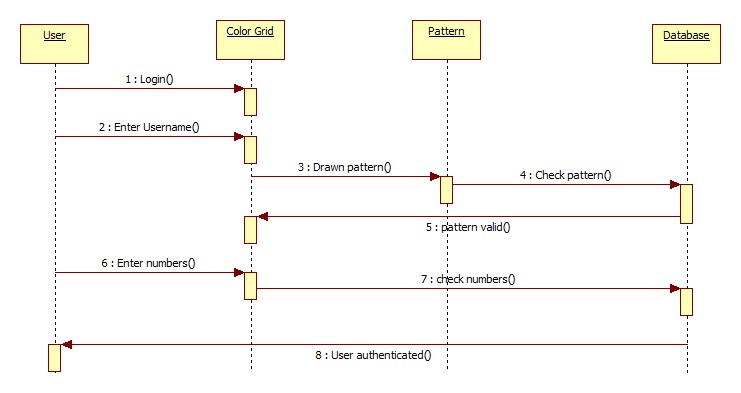
**Figure 6. Use Case Diagram for Login**

### **Sequence Diagram**

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.



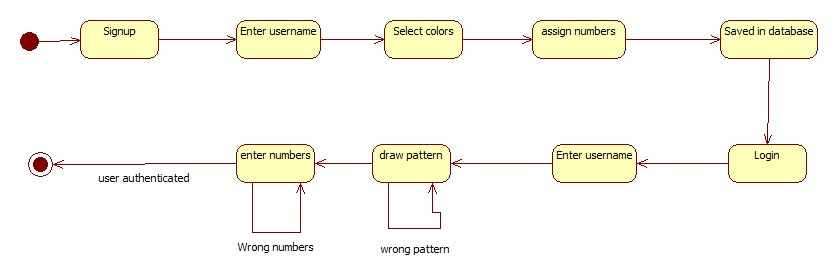
**Figure 7. Sequence Diagram for Signup**



**Figure 8. Sequence Diagram for Login**

### **State Diagram**

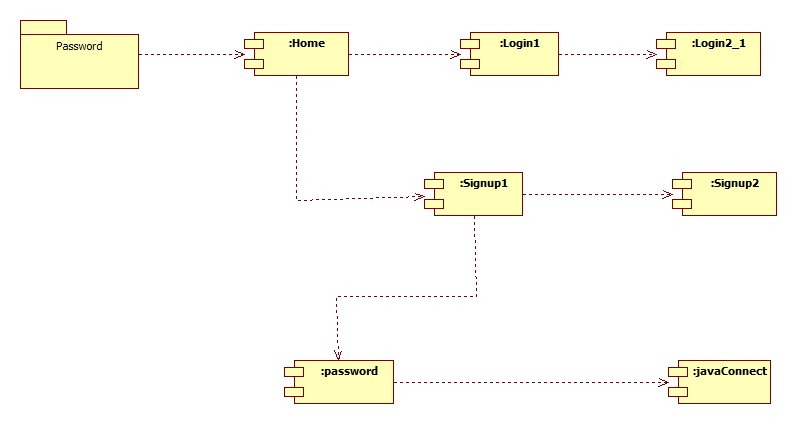
A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction.



**Figure 9. State Diagram**

### **Component Diagram**

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make that functionalities .So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.



**Figure 10. Component Diagram**

# Technologies Used

## **HTML**

HTML which is short for HyperText Markup language is a computer language used to create the web views for a website or simply known as HTML pages. A cluster of HTML pages create a whole website where internet users can browse through and get the required knowledge.

In this project the HTML pages are index.html, Login.html, Signup.html, There are different links or hyperlinks inside the pages which help the user to jump from one web view to another.

## **CSS**

As HTML is the structure of the webpage, CSS defines the design of the page, CSS which is short for Cascading style sheet is used for describing the presentation of the hypertext markup of the page. I gives the visual presentation for the website.

In this project the CSS styles applied is index.css there are external css styles applied in this project the external styles helps support the project externally by providing predefined styles. The external style used in the project is Twitter’s bootstrap.css.

## **JavaScript/Jquery**

Javascript is a high level definition language used to add functionality to the webpages/websites. Usually websites does more than what it looks like the process going on behind a web application is undertaken by javascript, The processing taken behind the website is usually done by Javascript.

In this project the javascript files are login.js and signup.js which helps adding click and mouse over functionality into the webpages apart from these two javascript files there are external javascript files added to the project likewise notify.js and alertify.js.

## **NodeJs**

Node.js is a server side platform which is build on the chrome’s javascript V8 engine. Node.js is an open source javascript framework which is usually used for writing the server side of a web site which usually manipulates and communicates with the database of the system.

## **MongoDb**

MongDb is a noSQL database. MongoDb stores data in a flexible JSON format, fields in this can vary from different document to document.

# METHODOLOGY:

A design and implementation program should also be integrated with the formal system development life cycle to include a business case, requirements definition, design, and implementation plans. Technology and design methods should be included, as well as the security processes necessary to provide the following services across all technology layers:

1. Authentication

2. Authorization

3. Availability

4. Confidentiality

5. Integrity

6. Accountability

7. Privacy.

## **HOVERING**

In [computing](http://en.wikipedia.org/wiki/Computing), a **mouseover**, **mouse hover** or [**hover box**](http://en.wikipedia.org/wiki/Hoverbox) is a [graphical control element](http://en.wikipedia.org/wiki/Graphical_control_element) that is activated when the user moves or "hovers" the [pointer](http://en.wikipedia.org/wiki/Pointer_(computing_WIMP)) over its trigger area, usually with a [mouse](http://en.wikipedia.org/wiki/Mouse_(computing)), but also possible using a [digital pen](http://en.wikipedia.org/wiki/Digital_pen).

Mouseover events are not limited to web design and are commonly used in modern GUI programming. Their existence might not even be known to the user as the events can be used to call any [function](http://en.wikipedia.org/wiki/Function_(programming)) and might affect only the internal workings of the program.

Mouse events notify when the user uses the mouse (or similar input device) to interact with a component. Mouse events occur when the cursor enters or exits a component's onscreen area and when the user presses or releases one of the mouse buttons.

Tracking the cursor's motion involves significantly more system overhead than tracking other mouse events. That is why mouse-motion events are separated into Mouse Motion listener type

public interface MouseListener extends[EventListener](https://docs.oracle.com/javase/8/docs/api/java/util/EventListener.html)

The listener interface for receiving "interesting" mouse events (press, release, click, enter, and exit) on a component. (To track mouse moves and mouse drags, use theMouseMotionListener.)

The class that is interested in processing a mouse event either implements this interface (and all the methods it contains) or extends the abstract MouseAdapter class (overriding only the methods of interest).

The listener object created from that class is then registered with a component using the component's addMouseListener method. A mouse event is generated when the mouse is pressed, released clicked (pressed and released). A mouse event is also generated when the mouse cursor enters or leaves a component. When a mouse event occurs, the relevant method in the listener object is invoked, and the MouseEvent is passed to it.

**The Mouse Listener API**

The MouseListener Interface

|  |  |
| --- | --- |
| **Method** | **Purpose** |
| mouseClicked(MouseEvent) | Called just after the user clicks the listened-to component. |
| [mouseEntered(MouseEvent)](https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseListener.html#mouseEntered-java.awt.event.MouseEvent-) | Called just after the cursor enters the bounds of the listened-to component. |
| [mouseExited(MouseEvent)](https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseListener.html#mouseExited-java.awt.event.MouseEvent-) | Called just after the cursor exits the bounds of the listened-to component. |
| [mousePressed(MouseEvent)](https://docs.oracle.com/javase/8/docs/api/java/awt/event/MouseListener.html#mousePressed-java.awt.event.MouseEvent-) | Called just after the user presses a mouse button while the cursor is over the listened-to component. |
| mouseReleased(MouseEvent) | Called just after the user releases a mouse button after a mouse press over the listened-to component. |

Table 6.1 Mouse listener API

## **Random**

By "random", we usually mean something like "unpredictable and unbiased". Or put another way, an ideal **random number generator** (RNG) would generate a series of numbers within a particular range where: each possible number is *equally likely* to occur on every iteration.

(hence) there is *no dependency* between successive numbers generated.

The way of generating random numbers in Java that most people know about is using the java.util.Random class.

notable cases where java.util.Random is not suitable include:

* **Simulations**
* Cases where you are generating a **large number of *combinations*** of numbers: e.g. in an application such as my LetterMeister game for iPhone, a large number of combinations of word positions must be generated and tested; another typical case would be "soak" testing many applications or routines;
* Cases where random numbers are part of any **security** feature of an application;
* Games where random numbers aren't simply a "casual" part of the game, notably **gambling** simulations.

public class **Random** extendsObject implementsSerializable

An instance of this class is used to generate a stream of pseudorandom numbers. The class uses a 48-bit seed, which is modified using a linear congruential formula.

If two instances of Random are created with the same seed, and the same sequence of method calls is made for each, they will generate and return identical sequences of numbers. In order to guarantee this property, particular algorithms are specified for the class Random. Java implementations must use all the algorithms shown here for the class Random, for the sake of absolute portability of Java code. However, subclasses of class Random are permitted to use other algorithms, so long as they adhere to the general contracts for all the methods.

The algorithms implemented by class Random use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Many applications will find the method [Math.random()](http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html" \l "random()) simpler to use.

Due to the security issues in random numbers we prefer shuffling the color grid for better security.

# Hybrid Textual Authentication Scheme

1. During registration, user should rate colors as shown in figure 1. The User should rate colors from 1 to 8 and he can remember it as “RLYOBGIP”. Same rating can be given to different colors.



Figure 6.1 Rating of colors

1. During the login phase, when the user enters his username an interface is displayed based on the colors selected by the user. The login interface consists of grid of size 8×8. This grid contains digits 1-8 placed randomly in grid cells. The interface also contains strips of colors as shown in the figure above. The color grid consists of 4 pairs of colors. Each pair of color represents the row and the column of the grid.



Table 6.2 login interface

Figure 6.2 shows the login interface having the color grid and number grid of 8 x 8 having numbers 1 to 8 randomly placed in the grid. Depending on the ratings given to colors, we get the session password. As discussed above, the first color of every pair in color grid represents row and second represents column of the number grid. The number in the intersection of throw and column of the grid is part of the session password. Consider the figure 9 ratings and figure 10 login interfaces for demonstration. The first pair has red and yellow colors. The red color rating is 1 and yellow color rating is 3. So the first letter of session password is 1st row and 3rd column intersecting element i.e. 3. The same method is followed for other pairs of colors. For figure 10 the password is “3573”. Instead of digits, alphabets can be used. For every login, both the number grid and the color grid get randomizes so the session password changes for every session.

# Resilience of proposed system against attacks

1. **Dictionary Attacks:** Graphical passwords are less vulnerable to dictionary attacks. In proposed system, as the user only select color and assign number to the selected color.
2. **Guessing Attacks:** Guessing attack is another eminent strategy used by the intruders. If the attacker tries on guessing the colors it would be of no use since the colors of the squares get changed for every login attempt. Hence the probability of guessing attacks is very low.
3. **Spyware attacks:** Excluding a few exception, key loggers and screen loggers cannot be used to attack against this method. By using a key logger if the attacker knows the colors of the squares , it would be of no use as in the second system attacker will be expected to enter the number assigned to him during registration .Hence our proposed system is resistant to spyware attacks.
4. **Shoulder surfing:** In the proposed system, even if the peeper observes the color of the square in which the password character lies, he cannot identify the sequence of colors. Thus proposed system is resilient to shoulder surfing attacks.
5. **Social Engineering:** Compared to ordinary alphanumeric passwords, it is inconvenient for a user to give his graphical password to another person. Hence graphical passwords are less susceptible to social engineering attacks.

# IMPLEMENTATION DETAILS

## **Workflow Diagram**

Many methodology or findings from this field mainly generated into journal for others to

take advantages and improve as upcoming studies. The method is use to achieve the

objective of the project that will accomplish a perfect result. In order to evaluate this

project, the methodology based on System Development Life Cycle (SDLC), generally

three major step, which is planning, implementing and analysis.

The final year project used three major steps to implement project starting from planning,

Implementing and testing all the methods used for finding and analysing data regarding

the project related.

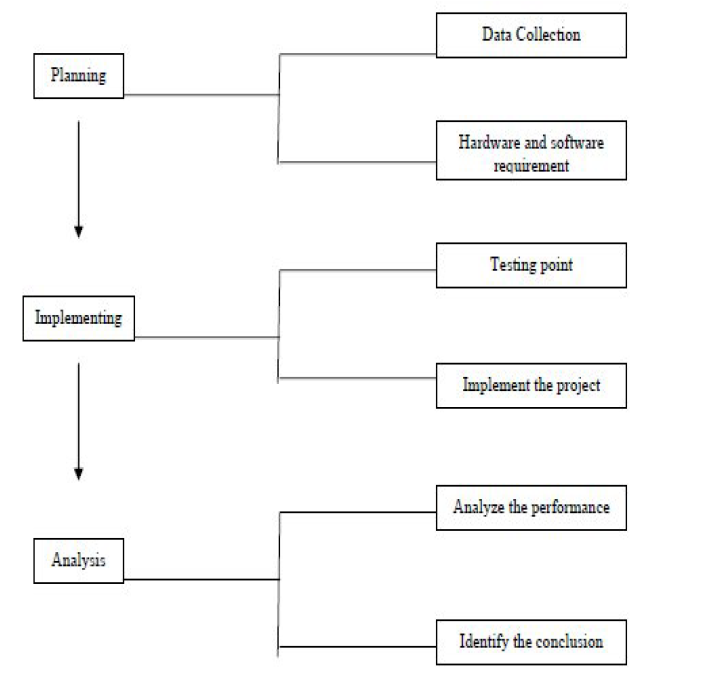


Fig 7.1 Work Flow Diagram

# Screenshot

## **Home Page**

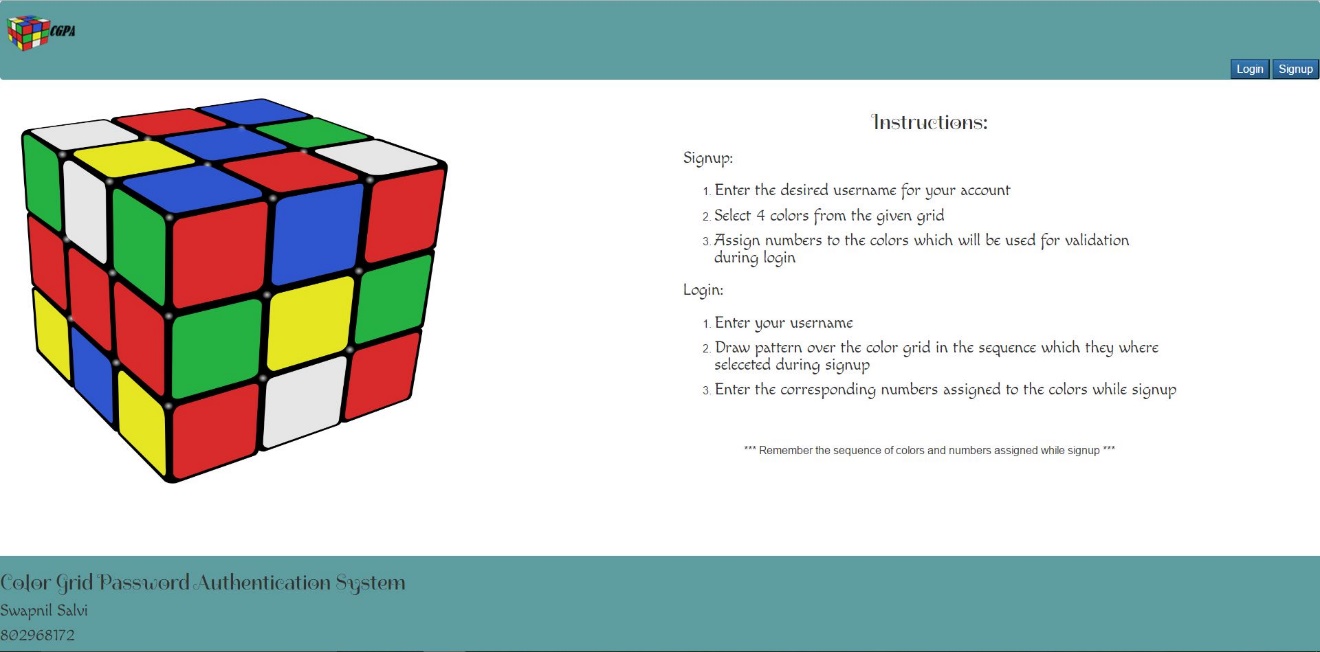
****

Fig 7.2.1 Screenshot of home

## **Enter username for Signup.**

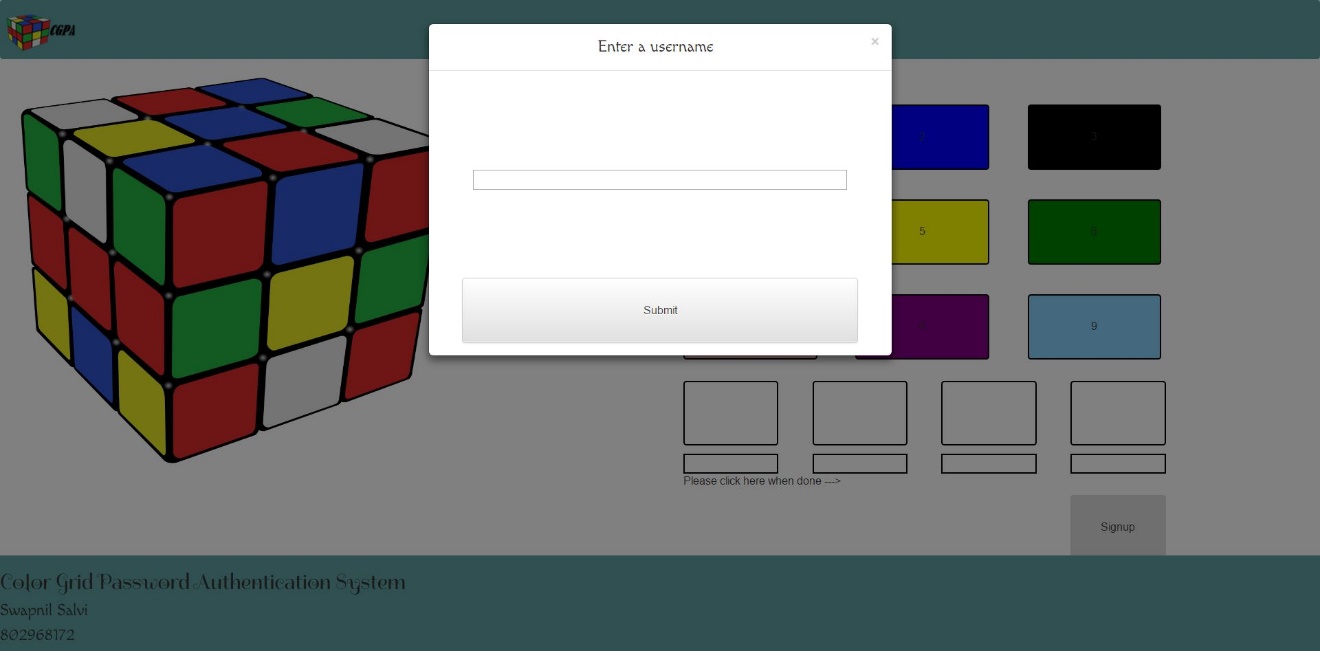


Fig 7.2.3 Screenshot of username for Signup.

## **Select any four colors from the grid**

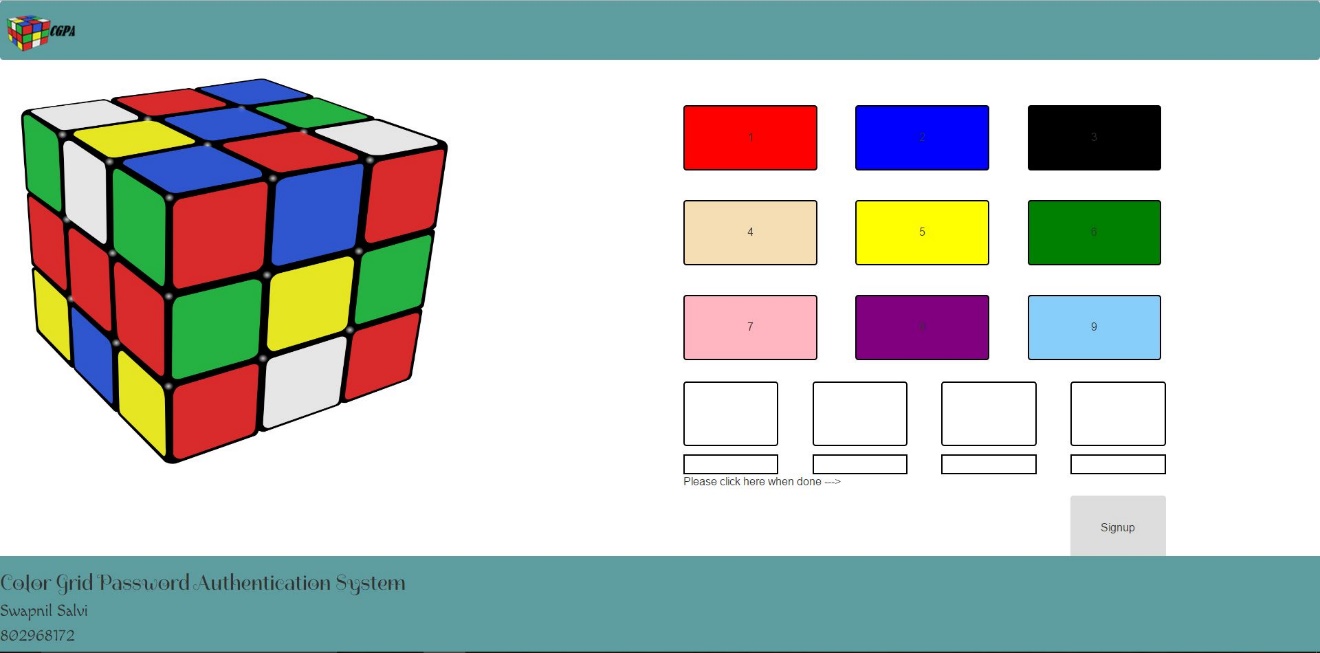
****

Fig 7.2.3 Screenshot of selecting four colors

## **After selecting four colors**

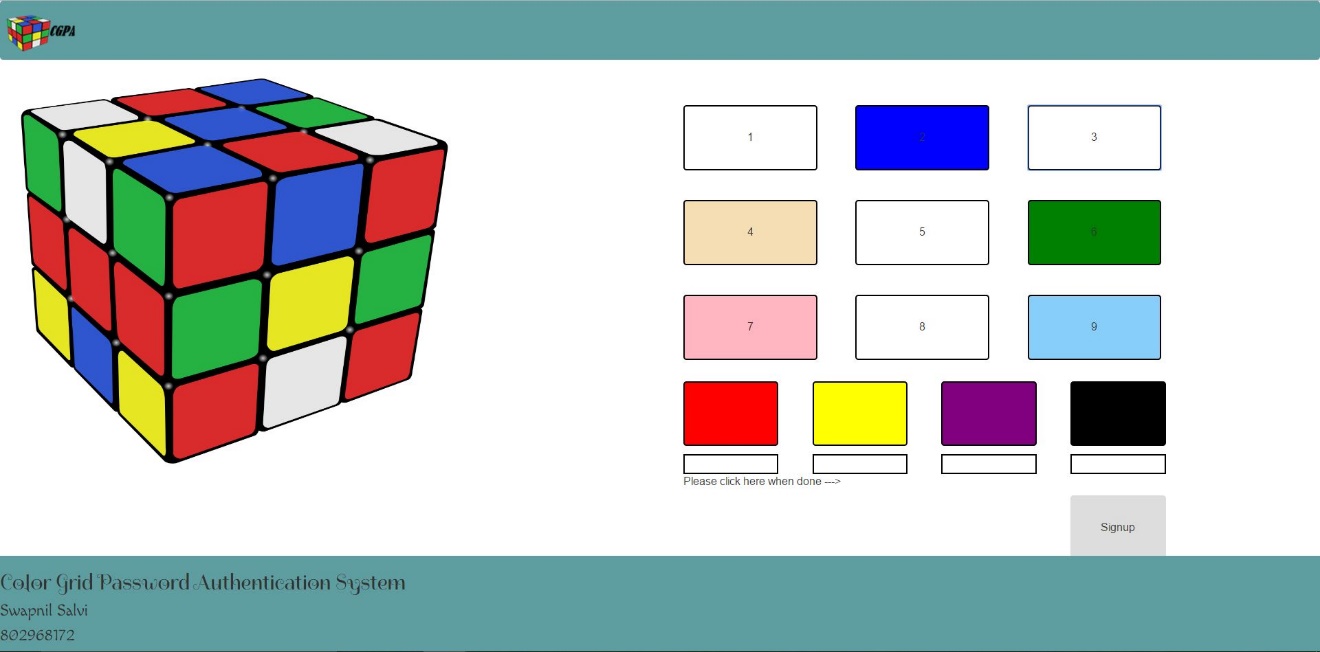


Fig 7.2.4 Screenshot of after selecting four colors

## **Assign numbers to that colors (insert only single value)**

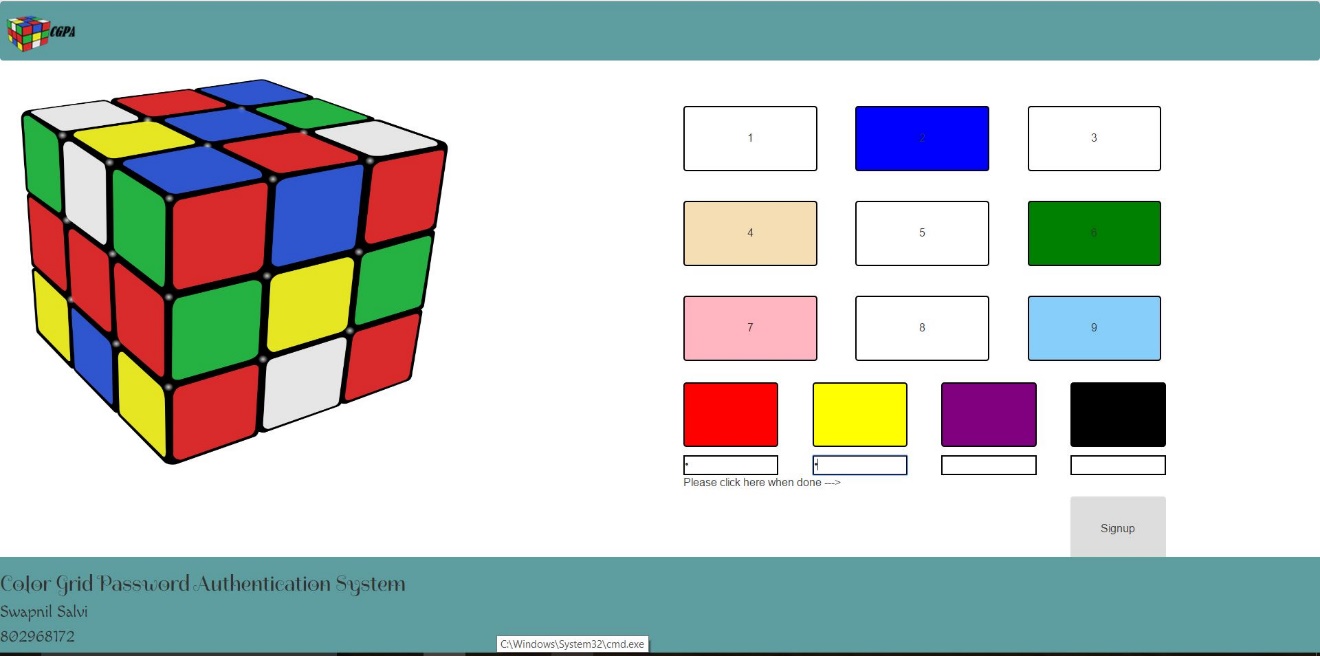
****

Fig 7.2.5 Screenshot of assigning numbers to colors

## **Save and signup successful**

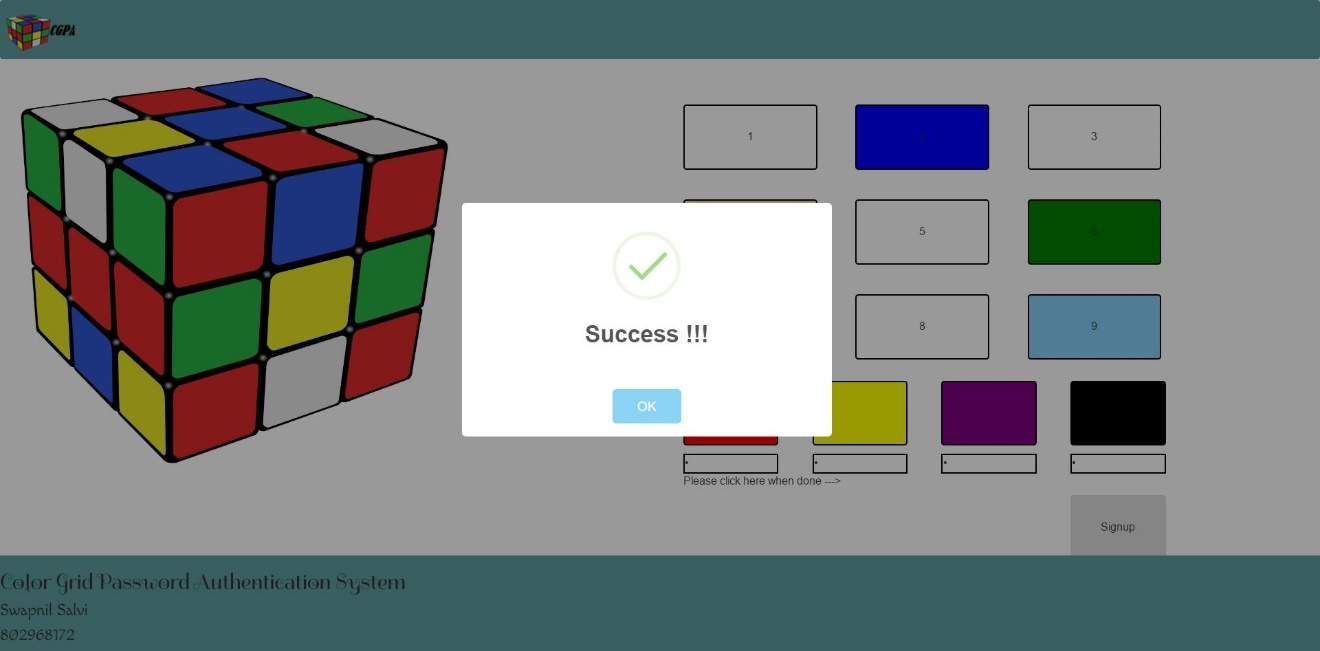


Fig 7.2.6 Screenshot of signup Successful

## **Draw pattern on the grid of selected colors**

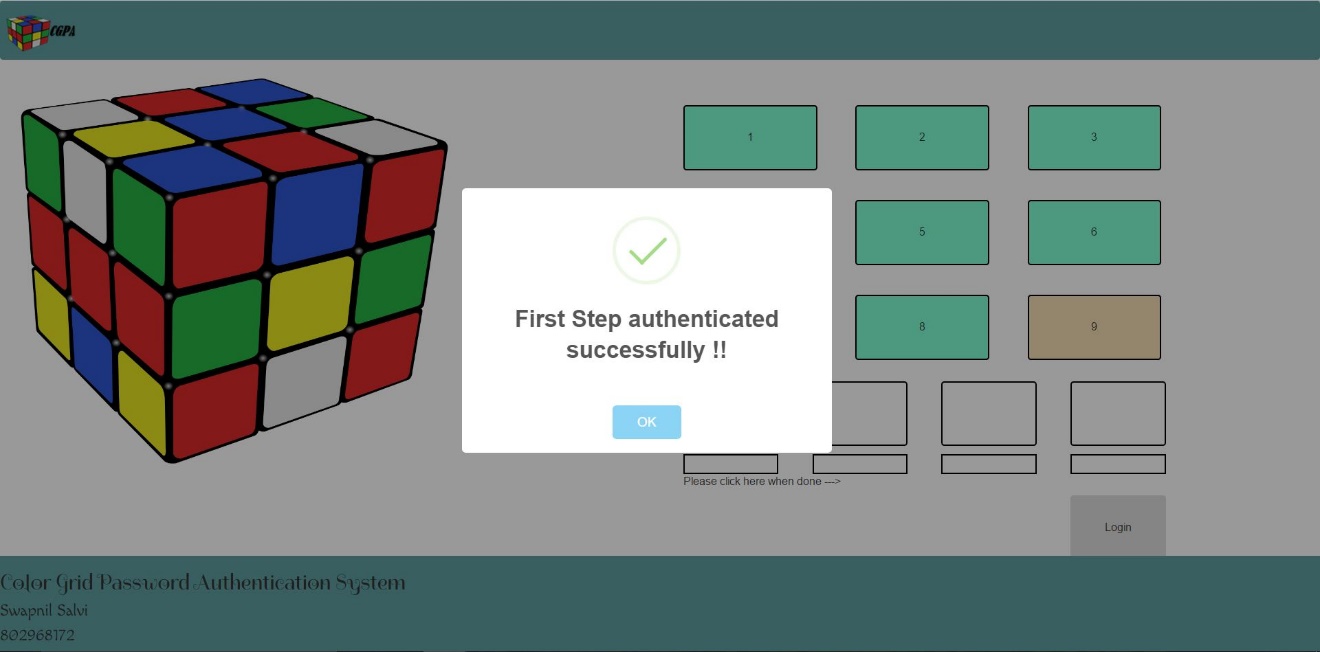
****

Fig 7.2.7 Screenshot of drawing pattern

## **Draw valid pattern of selected colors**

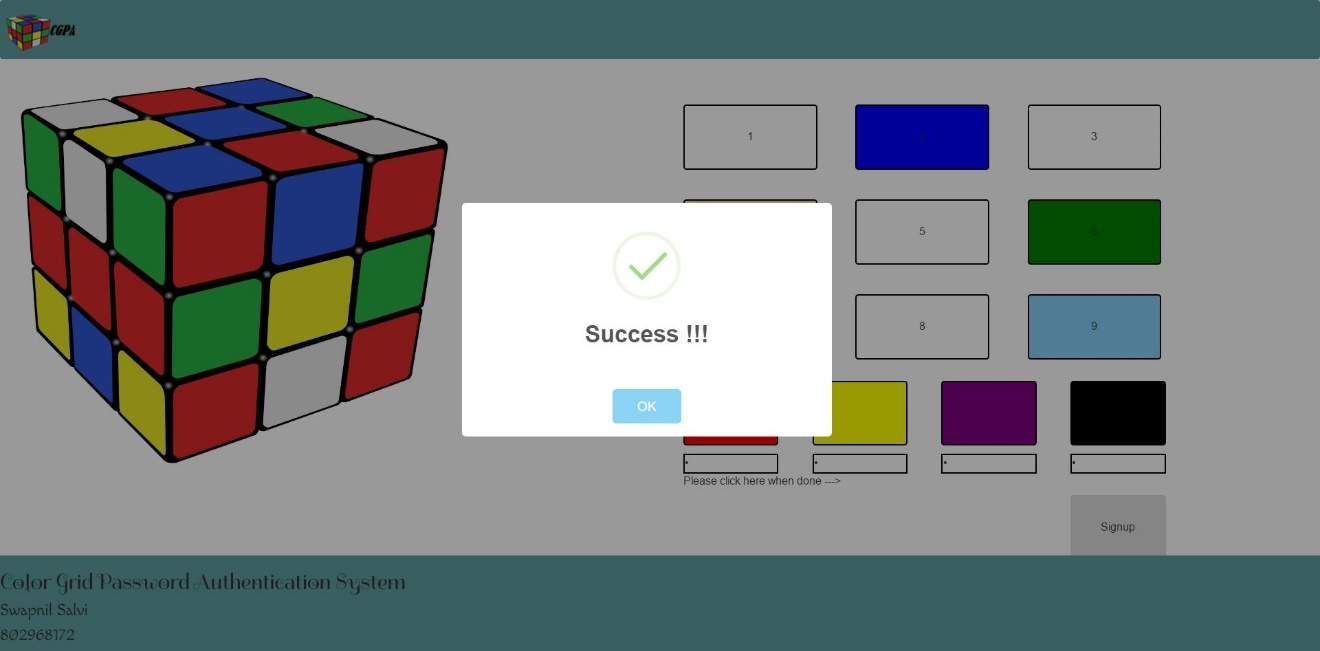


Fig 7.2.8 Screenshot of valid path

## **Enter Pin for selected colors.**

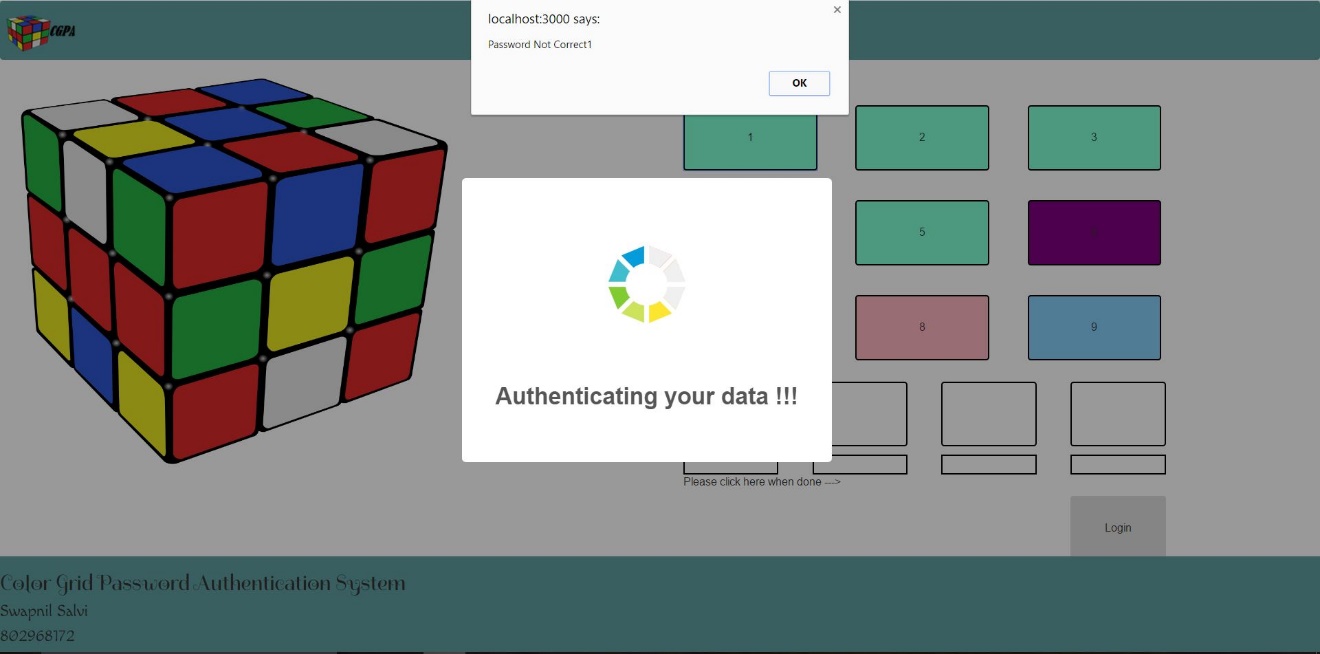
****

Fig 7.2.9 Screenshot of entering pin to the colors.

## **Login successful**

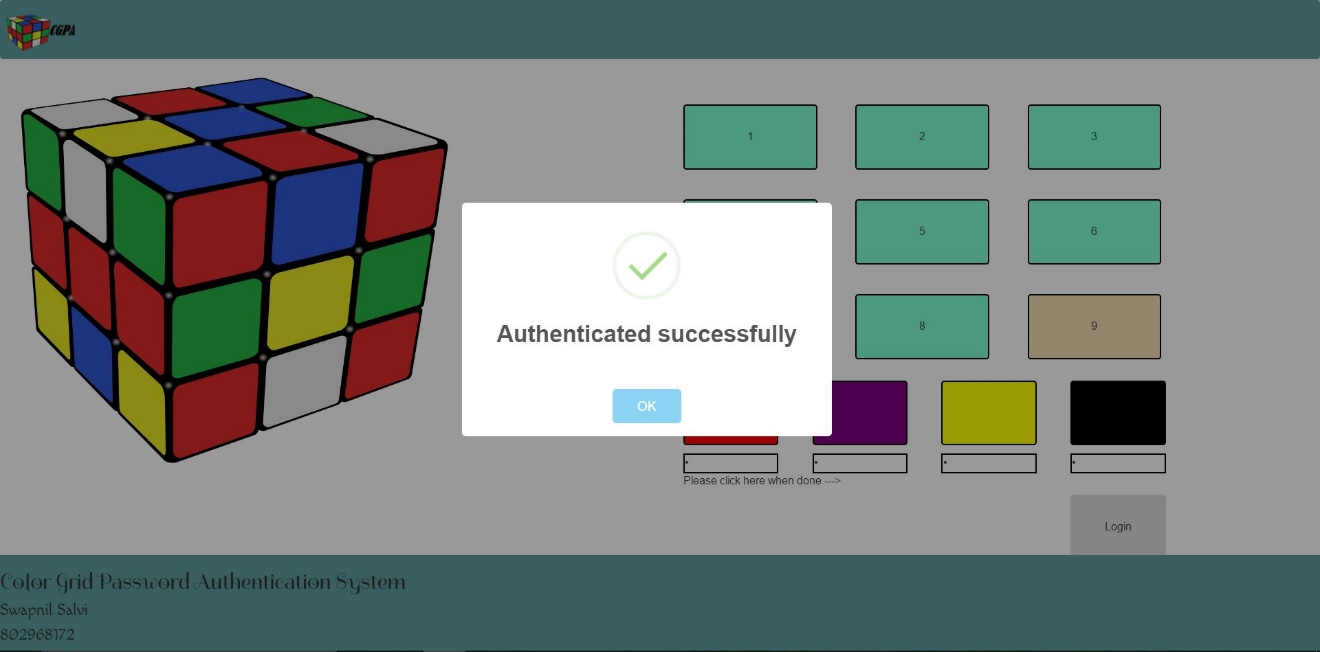
****

Fig 7.2.10 Screenshot of login successful.

# TimeLine Chart

A Gantt chart is a type of bar chart, developed by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e. precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line as shown below.

Although now regarded as a common charting technique, Gantt charts were considered extremely revolutionary when first introduced. This chart is also used in information technology to represent data that has been collected.

Table 7.1 Gantt Chart

# Project time line and task distribution

This is the table that involves the various activities involved and the time estimated per activity.

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity**  **Number.** | **Activities** | **Must be preceded by** | **Estimated Time Days** |
| A | Collected information about the problem domain. | - | 23 |
| B | Study of pattern matching ,random numbers | - | 15 |
| C | Analyzed the requirements for the desired application. | A | 12 |
| D | Module specification of proposed system | C | 10 |
| E | Carried out structural modeling with flowcharts, use case diagrams. | D | 18 |
| F | Development .  Coding: java modules | E | 40 |
| G | System Development/ Coding : /GUI design/ database System | F | 20 |
| H | Testing | G | 8 |
| I | Final Documentation | H,G | 5 |
| J | Development | I | 29 |
| K | Testing | J | 10 |
| L | Implementation | K | 4 |
| M | Final Documentation | K | 10 |

# Software testing

The testing phase involves the testing of the developed system using various kinds of data. An elaborated testing of data is prepared and a system is tested using the test data. While testing, errors are noted and corrections remade, the corrections are also noted for future use.

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

* 1. **Stress Testing**  
     is a type of performance testing, in which software is subjected to peak loads and even to a break point to observe how the software would behave at breakpoint. Stress testing also tests the behavior of the software with insufficient resources like CPU, Memory, Network bandwidth, Disk space etc. Stress testing enables to check some of the quality attributes like robustness and reliability.
  2. **User Acceptance testing (UAT )**  
     User Acceptance testing is a must for any project; it is performed by clients/end users of the software. User Acceptance testing allows SMEs (Subject matter experts) from client to test the software with their actual business or real-world scenarios and to check if the software meets their business requirements.
  3. **Vulnerability Testing**  
     involves identifying, exposing the software Vulnerabilities that can be exploited by hackers and other malicious programs likes viruses or worms. Vulnerability Testing is key to software security and availability. With increase number of hackers and malicious programs, Vulnerability Testing is critical for success of a Business
  4. **System Testing**  
     this includes multiple software testing types that will enable to validate the software as a whole (software, hardware and network) against the requirements for which it was built. Different types of tests (GUI testing, Functional testing, Regression testing, Smoke testing, load testing, stress testing, security testing, stress testing, ad-hoc testing etc.,) are carried out to complete system testing.
  5. **Smoke testing**  
     is a type of testing that is carried out by software testers to check if the new build provided by development team is stable enough i.e., major functionality is working as expected in order to carry out further or detailed testing. Smoke testing is intended to find “show stopper” defects that can prevent testers from testing the application in detail. Smoke testing carried out for a build is also known as build verification test.
  6. **Security Testing**  
     is a type of software testing carried out by specialized team of software testers. Objective of security testing is to secure the software is to external or internal threats from humans and malicious programs. Security testing basically checks, how good is software’s authorization mechanism, how strong is authentication, how software maintains confidentiality of the data, how does the software maintain integrity of the data, what is the availability of the software in an event of an attack on the software by hackers and malicious programs is for Security testing requires good knowledge of application, technology, networking, security testing tools. With increasing number of web applications necessarily of security testing has increased to a greater extent.
  7. **Acceptance Testing**  
     Acceptance testing is a formal type of software testing that is performed by end user when the features have been delivered by developers. The aim of this testing is to check if the software confirms to their business needs and to the requirements provided earlier. Acceptance tests are normally documented at the beginning of the sprint (in agile) and is a means for testers and developers to work towards a common understanding and shared business domain knowledge
  8. **Beta Testing**  
     This is a formal type of software testing that is carried out by end customers before releasing or handing over software to end users. Successful completion of Beta testing means customer acceptance of the software.
  9. **Black Box testing**  
     Black box testing is a software testing method where in testers are not required to know coding or internal structure of the software. Black box testing method relies on testing software with various inputs and validating results against expected output.
  10. **GUI (Graphical User Interface) testing**  
      This type of software testing is aimed at testing the software GUI (Graphical User Interface) of the software meets the requirements as mentioned in the GUI mockups and Detailed designed documents. For e.g. checking the length and capacity of the input fields provided on the form, type of input field provided, e.g. some of the form fields can be displayed as dropdown box or a set of radio buttons. So GUI testing ensures GUI elements of the software are as per approved GUI mockups, detailed design documents and functional requirements.
  11. **Glass box Testing**  
      Glass box testing is another name for White box testing. Glass box testing is a testing method that involves testing individual statements, functions etc., Unit testing is one of the Glass box testing methods
  12. **Integration Testing**  
      Integration testing also known as I&T in short, in one of the important types of software testing. Once the individual units or components are tested by developers as working then testing team will run tests that will test the connectivity among these units/component or multiple units/components. There are different approaches for Integration testing namely, Top-down integration testing, Bottom-up integration testing and a combination of these two known as Sand witch testing.
  13. **TEST CASES.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case No.** | **Test Condition** | **Operator Action** | **Input Specification** | **Output Specification(Expected Results)** | **Pass or Fail** |
| 1 | Sign up | Enter Username | Already Exists | Prompts an alert message | Pass |
| 2 | Signup | Enter Username | Does not Exist | Load the Color Grid form | Pass |
| 3 | Login | Enter Username | Valid Username Entered | Load the Color Grid form for further Authentication | Pass |
| 4 | Login | Enter Username | Invalid Username Entered | Prompts an error message and load the login page | Pass |
| 5 | Color Grid | Draw Pattern | Valid Pattern entered according to the sequence of colors selected during signup | Enter the corresponding numbers assigned to the colors during signup | Pass |
| **Test Case No.** | **Test Condition** | **Operator Action** | **Input Sepecification** | **Output Specification(Expected Results)** | **Pass or Fail** |
| 6 | Color Grid | Draw Pattern | Invalid Pattern entered | Shuffle the grid and loads again. | Pass |
| 7 | Numbers | Enter numbers | Valid numbers corresponding to the colors are entered | User Authenticated | Pass |
| 8 | Numbers | Enter numbers | Invalid numbers entered | Reloads the color grid form | Pass |

# Conclusion

The concept of color grid password authentication provides a brand new opportunity for the development of security in applications since it overcomes many attacks and makes highly secure environment for the application. A application needs a constant security that might prove to be an Achilles heel for the security domain. The two schemes. Hybrid textual password authentication scheme and pattern based password authentication scheme. The project will also implement them individually to compare them. Then it will integrate the two above mentioned password schemes. Both the authentication schemes were vulnerable to many attacks which can be overcome by integrating both of them and using them on a single standalone application.

# REFERENCE:

[1]http://www.ijcset.com/docs/IJCSET13-04-04-083.pdf

[2]http://www.ijarcsse.com/docs/papers/Volume\_4/4\_April2014/V4I4-0456.pdf

[3]http://www.ijser.org/researchpaper%5CAuthentication-schemes-for-session-password.pdf

[4] http://cups.cs.cmu.edu/soups/2011/proceedings/a6\_Zakaria.pdf

[5] http://people.scs.carleton.ca/~paulv/papers/exploiting\_clickbased.pdf

[6] (IJCSIS) International Journal of Computer Science and Information Security, Vol. 6, No. 2, 2009

[7] International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064 Impact Factor (2012): 3.358

[8] http://www.ijcset.com/ I**nternational Journal of Computer Science & Engineering Technology (IJCSET)**

[9] http://www.ijcsit.com/ International Journal of Computer Science and Information Technologies (IJCSIT®)

[10] International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.3, May 2011