Yusuf Koz

Avoiding Shoulder Surfing via Graphical Passwords

# **1. Summary of Proposal**

# **1.1 Aims and Objectives**

# **1.1.1 Aims**

# To design and implement a graphical password system, **ColorPass**, that dynamically resists shoulder surfing by incorporating randomized elements, such as changing colors or character positions after each selection.

# **1.1.2 Objectives**

# Develop a user-friendly system that dynamically rearranges the color grid after each input.

# Build a secure backend to handle password validation and storage.

# Test the system’s resistance to observation attacks and ensure it meets usability requirements.

# Provide accessibility options for users with visual impairments, such as patterns or alternate cues for colors.

# **3. Development Approach**

# **3.4 Adaptation of ColorPass Technique**

# The **ColorPass** method combines password characters with user-selected colors, adding an additional layer of security by dynamically changing the position of letters and colors **after each user selection during login**. This makes it virtually impossible for an observer to deduce the user’s password even if they watch the process multiple times.

# **How It Works:**

# During **registration**, the user selects a password (e.g., "ABC") and a specific color (e.g., "Red").

# During **login**, a grid of randomized characters with different colors is displayed. The user identifies their password characters based on their selected color and clicks the correct one.

# After each selection, the grid **refreshes**, and:

# The characters are rearranged into new positions.

# The colors assigned to characters are randomized again.

# **Security Benefit:** This dynamic behavior ensures that even if an observer watches the entire login process, they cannot infer the correct password or its color mapping because it changes unpredictably after each input.

# **5. Technical Specifications and System Design**

# **5.1 Frontend (User Interface)**

# **Technologies**

# **HTML**: Structuring the registration and login pages.

# **CSS**: Styling dynamic grids for clear visibility of colors and characters.

# **JavaScript (React or Vue.js):** Handling real-time interactions, such as refreshing the grid dynamically after each user selection.

# **Features**

# **Registration Page:**

# Input fields for password creation.

# Dropdown or clickable options for color selection.

# **Login Page:**

# A **dynamic grid** where each cell contains:

# A letter (e.g., A, B, C).

# A randomly assigned color.

# After each selection:

# The characters are reshuffled into new grid positions.

# The colors are reassigned to new characters randomly.

# The system validates the user’s clicks in sequence against the stored password and color.

# **5.2 Backend (Server and Database)**

# **Technologies**

# **Python (Flask/Django)** or **Java (Spring Boot)** for server logic.

# **PostgreSQL/MySQL** for secure database storage.

# **Features**

# **Dynamic Grid Logic**:

# On each character input, the server generates a new randomized grid.

# Sends the updated grid back to the frontend for display.

# **Login Validation**:

# Matches the user’s selected sequence of characters and their associated colors against the stored password and color.

# If the sequence is correct, the login is successful. Otherwise, an error is returned.

# **5.3 How the Dynamic Behavior Works**

# **Initial Grid Generation:**

# When the user accesses the login page, a grid of randomized characters is generated.

# Each character is displayed with multiple colors.

# **After a Selection:**

# When the user clicks on the correct character based on their color:

# The **frontend sends the selected input** to the backend.

# The **backend verifies** the input and triggers the next step.

# A **new grid** is generated with:

# Characters reshuffled randomly across the grid.

# New colors assigned to each character.

# **Repeat Until Completion:**

# The process repeats for each character in the password until the entire sequence is verified.

# **Security Advantage**

# Each interaction creates a **unique grid**. Even if a malicious observer memorizes part of the grid, it becomes irrelevant after the next shuffle.

# **5.4 Database Design**

# **Table: Users**

| **Column Name** | **Data Type** | **Description** |
| --- | --- | --- |
| user\_id | Integer | Unique identifier for each user. |
| username | String | User’s unique username. |
| password | String | Hashed and salted password. |
| selected\_color | String | User’s chosen color. |
| created\_at | Timestamp | Date and time of account creation. |

# **Additional Backend Logic**

# **Rate Limiting:** To prevent brute-force attacks, limit login attempts.

# **Session Tokens:** Issue secure session tokens after successful login.

# **6. Timeline and Key Phases**

# **6.1 Phase 1: Requirements Gathering and Initial Development (Weeks 1–6)**

# Research and define functional requirements.

# Develop the dynamic grid generation logic.

# Create frontend wireframes and backend API endpoints.

# **6.2 Phase 2: Implementation and Testing (Weeks 7–12)**

# Build and integrate the frontend and backend.

# Test the dynamic grid behavior for performance and accuracy.

# Conduct usability testing with users to refine the interface.

# Perform security testing to ensure resilience to observation attacks.

# **7. Challenges and Risk Mitigation**

# **7.1 Dynamic Grid Complexity**

# **Risk:** Frequent reshuffling might confuse users. **Solution:** Use subtle animations and clear instructions to guide users.

# **7.2 Security**

# **Risk:** Unauthorized access to the password logic. **Solution:** Encrypt all communication using HTTPS and store passwords securely with hashing.

# **7.3 Scalability**

# **Risk:** High computational cost of generating grids dynamically. **Solution:** Use efficient algorithms for randomization and caching techniques for high traffic.

# **8. Learning Outcomes and End-Product**

# **8.1 Key Skills Developed**

# Implementation of dynamic user interfaces using modern JavaScript frameworks.

# Secure backend design with industry-standard password storage and API endpoints.

# Testing and optimization of dynamic features for usability and security.

# **8.2 Description of Final Prototype**

# The **ColorPass** system will:

# Allow users to register with a password and color of their choice.

# Present a **dynamic login experience** where:

# A grid of randomized characters and colors is refreshed after each selection.

# Inputs are validated against the stored credentials.

# Be tested and refined for security (shoulder-surfing resistance) and usability.