Last Edit: February 8, 2025

Jordan Bassett

Lizbeth Garcia-Lopez

Password Factory

* **Premise**: Kids work in a password factory, where they must create the strongest passwords to keep data safe. Child slavery is back baby!
* **How it works**:
  + Players "assemble" passwords using different characters, numbers, and symbols to make the strongest possible password.
  + They can get tips on best practices like mixing uppercase, lowercase, and symbols.
  + Players face time challenges where they have to quickly create passwords before a “security breach” happens
  + Have prompts while they are working, like “use a special character” “More letters” etc
  + +points for getting a round right
  + Traumatize the child if they fail
  + The game might also have a "password lab" where kids test the strength of passwords and learn what works and what doesn't. This is also a backdoor for the devs (us) to just test things without playing the game

# Explanation of the Program:

1. **Password Generation**: The generate\_password function creates a random password using a mix of letters, numbers, and symbols, making it strong by default. While this is not actively used in the main challenge, it could be expanded to provide random suggestions or even an automatic password creation feature.
2. **Password Strength Check**: The check\_password\_strength function evaluates the password based on length, the presence of lowercase and uppercase letters, digits, and symbols. The game provides feedback on whether the password is strong, moderate, or weak.
3. **Password Assembly Challenge**:
   * The player is given a X-second challenge to create a password.
   * After the player inputs their password, it's checked for strength, and the program gives feedback.
   * A simulated "security breach" is tested by waiting 5 seconds, and if the password is strong, the player gets a positive result.
   * If its wrong, we use our new UI skills to flash the screen with a bombardment of annoying colors. Also we notify the authorities.
4. **Replay Option**: After each challenge, the player can choose to play again or exit the game.

# How to Play:

1. When the game starts, the player is prompted to create a password within 30 seconds.
2. The program checks if the password meets the criteria for strength and provides feedback.
3. It then simulates a security breach and checks if the password could withstand it.
4. The player can try again if they want.

# Potential Improvements:

* **Score Tracking**: You could add a score or reward system based on how fast the player creates strong passwords.
* **Multiple Levels**: You could add difficulty levels with different password complexity requirements.
* **Password Lab**: A feature where the player can test various passwords against common attacks like brute force or dictionary-based guesses.

# Program Flow

**1. Start Screen (Main Menu)**

* Display game title and simple instructions.
* Options:
  + **Start Game**
  + **Password Lab (optional)**
  + **Exit**

**2. Game Start (Password Assembly Challenge)**

* Display password-building interface.
* Timer (X seconds) starts.
* Player assembles a password using different characters (letters, numbers, symbols).
* Player submits the password.

**3. Password Evaluation**

* Check password strength based on:
  + Length (minimum 8 characters)
  + Presence of uppercase, lowercase, numbers, symbols
  + Avoidance of common dictionary words
* Provide feedback (Weak, Moderate, Strong).

**4. Security Breach Simulation**

* If the password is **Weak**, an instant failure message appears. Chastise the child
* If **Moderate**, a message warns that it might be crackable. Chastise the child
* If **Strong**, a positive reinforcement message appears. Notify the FBI of a potential threat the national security.
* Display an animation or text showing whether the password "withstood" the attack.

**5. Score & Replay Option**

* Display score based on strength & speed of password creation.
* Option to:
  + Try again
  + Return to main menu
  + Quit

**(Optional) Password Lab**

* A free-play mode where players can enter passwords to test their strength without a timer.
* The game explains why a password is strong or weak.
* This would just be the password strength check without having to play the game, good for testing

## Basic Program Flow Diagram

START -> Main Menu -> Select Option

|

|--> Start Game --> Password Assembly (30s timer) --> Submit Password

|

|--> Evaluate Strength (Weak / Moderate / Strong)

|

|--> Security Breach Test (Success or Fail)

|

|--> Score & Replay Option (Replay / Main Menu / Exit)

|--> Password Lab (Test passwords freely, get feedback)

|--> Exit

Since checking against a full dictionary is outside our skill set (and overkill) for a kids' game), We can

* Focus on structural rules instead of dictionary-based validation.
* Implement rules like:
  + Length requirement (e.g., at least 8 characters).
  + Must contain uppercase, lowercase, numbers, and symbols.
  + Avoid simple patterns (e.g., "123456" or "aaaaaa").
  + Penalty for repetition (e.g., "passwordpassword").
  + Encourage diversity (e.g., not all numbers or all letters).
  + Most real words follow a pattern of alternating vowels and consonants (e.g., “Dinosaur” = CVCVCVCV). Would take some research:

**Implementation Approach**

* Use regular expressions or simple loops to check conditions.
* Assign scores based on strength factors.
* Provide feedback messages to guide kids toward stronger passwords.

**Example Algorithm Outline**

1. Check length: If <8, fail immediately.
2. Check character diversity:
   * Has at least one uppercase? ✅
   * Has at least one lowercase? ✅
   * Has at least one number? ✅
   * Has at least one special character? ✅
3. Check for common patterns (optional):
   * Reject sequences like "1234", "aaaa", or "password".
4. Give a rating: Weak / Moderate / Strong.

**Game Flow After Algorithm is Done**

* Once have this is working, the rest is just user experience:
  + GUI Elements: A fun way to enter passwords.
  + Feedback UI: Make weak passwords “explode”
  + Instructions had emphasis on GUI
  + Can use a CSS Stylesheet for uniform look, and easy to edit everything at once

# Theme

* No theme, just make it work
* Dystopian Child Labor Password Factory
* Mean: Fail to meet the password standards, and the consequences are *immediate and absurd*—bees, alarms, and a polite but sternly worded letter from the FBI.+

# UI

**Password Factory - UI Design Template**

**1. Main Menu Screen**

**Components:**

* **Title:** "Password Factory" (Centered, Bold, Large Font)
* **Buttons:**
  + **Play the Game** → Goes to password entry/game screen
  + **Password Lab** → Opens a free-play password strength tester
  + **Exit** → Closes the program

**Design Notes:**

* Background color: Light Blue
* Buttons: Large, with rounded edges
* Font: Sans-serif, playful but clear
* Logo or header image (optional)

**2. Play the Game Screen**

**Components:**

* **Title:** "Enter a Strong Password"
* **Input Field:** Text box for entering a password
* **Submit Button:** "Check Password Strength"
* **Feedback Label:** Displays feedback (Weak, Moderate, Strong)
* **Back Button:** Returns to the Main Menu

**Design Notes:**

* Background: Light Gray or White
* Input Field: Large, centered
* Button: Bright color to encourage interaction
* Optional: Animated text or icons for feedback

**3. Password Lab Screen**

**Components:**

* **Title:** "Test Your Password"
* **Input Field:** Text box for entering a password
* **Check Strength Button:** Analyzes the password
* **Feedback Label:** Displays "Weak/Moderate/Strong" based on criteria
* **Back Button:** Returns to the Main Menu

**Design Notes:**

* Similar layout to "Play the Game" for consistency
* Additional tooltip or info icon explaining password strength factors

**4. Additional UI Notes**

* **Button Consistency:** All buttons should be similar in size and shape.
* **Navigation Clarity:** Every screen should have a "Back" button.
* **Animations (Optional):** Light transitions when switching screens.
* **Fonts & Colors:** Keep readable, kid-friendly, and accessible.

**Deliverables**

✅ Sketch or mock up each screen in **Word, PowerPoint, or pen & paper** ✅ Label all buttons and fields so implementation is clear ✅ Suggest color schemes or styling preferences

UML

 **App.java** **depends on** (and instantiates) **PasswordGame** and **PasswordLab**.

 **App.java** is **not the owner** of these screens, but it **controls the flow** by calling methods to display these screens.