Project 2: Password checker testing

Brett Worley

version: October 31, 2015

Contents

1 Introduction

The program to be developed is a password checking module. It will be used to check if a password meets the requirements of the system shown in the requirements section. This document covers the various testing strategies used to test the program. The finalized test plan was developed by using Equivalence class and boundary testing, Orthogonal array testing, and Decision table testing. Along with the code, a testing harness was created to provide automated testing and manual testing to check the coverage of the test cases.

1.1 Requirements

The requirements of the password checking program are as follows.

1. The new password shall be at least 9 characters long, and no longer than 24 characters.
2. New passwords cannot contain any blank spaces, and must contain only numberals, upper-case and lower-case letters, and the specials characters listed in Requirement 4.
3. All new passwords must contain at least two upper case letters, at least two lower case letters, and at least two numerals.

4. New passwords must contain at least two special characters from the following list: ! @ # $ % & *) ( ] [} { > < ; : . , / | \˜ ? _ − + =

1.2 Automated testing format

5. New passwords cannot be similar to any of the other two previous passwords, with "similar to" defined as "containing an identical five-character substring (either forward or backward), independent of letter case (for example, a5%Km and a5%kM would be considered identical substrings).

2 Equivalence class and boundary testing

2.1 Introduction

To implement equivalence classes and boundary testing, we will need to analyze the requirements for the system and develop equivalence classes that will show the border values to test against. By going through the requirements for the system, we can identify the border conditions and come up with the following equivalence partitions shown in table 1. The border conditions are important, as they are the most likely place that the system will fail on and around. Having these cases also enables testers to trim down similar or equivalent test cases that essentially test the same values, compared against the border conditions.

Table 1: Equivalence Partitions.

Less than 9 characters	At least 9 characters and not more than 24 characters	More than 24 characters
Contains no blank spaces	Contains no blank spaces	
Contains insufficient lower case	Contains sufficient lower case	
Contains insufficient upper case	Contains sufficient upper case	
Contains insufficient numbers	Contains sufficient numbers	
Contains insufficient special characters	Contains sufficient special characters	
Contains a identical five-character substring	Does not contain a indentical five-character subtring	

With these equivalence partitions, we are able to generate simple test cases based on passing or failing the classes one at a time. The following test plan is made up of 8 test cases that are designed to fail a certain requirement of the system.

Equivalence class and Boundary Test Plan

Test Case 1	
Purpose	Testing a valid password
Input	DahatB2559_@
Previous Passwords	ToT86635ss/< AVery990#^ #558#&;DoGs
Expected Output	ACCEPTED

Test Case 2	
Purpose	Testing against a similar password
Input	SsaPmis628@@
Previous Passwords	#558#&;DoGs AVery990#^ SimPass12!!
Expected Output	REJECTED: password too similar to a previous password

Test Case 3	
Purpose	Testing against a password containing a space
Input	&^!aaCH91 chat
Previous Passwords	#558#&;DoGs AVery990#^ ToT86635ss/<
Expected Output	REJECTED: password contains a space

Test Case 4	
Purpose	Testing against a password with not enough lower case letters
Input	#$678123HOUSE
Previous Passwords	ToT86635ss/< AVery990#^ SimPass12!!
Expected Output	REJECTED: password does not have enough lower case letters

Test Case 5	
Purpose	Testing against password with not enough upper case letters
Input	lowercasepasswords!;123
Previous Passwords	#558#&;DoGs AVery990#^ ToT86635ss/<
Expected Output	REJECTED: password does not have enough upper case letters

Test Case 6	
Purpose	Testing against not enough numbers
Input	NumberBoycott_?
Previous Passwords	#558#&;DoGs GUha891)) SimPass12!!
Expected Output	REJECTED: password does not have enough numbers

Test Case 7	
Purpose	Testing against a short password
Input	Do12ah_=
Previous Passwords	#558#&;DoGs GUha891)) SimPass12!!
Expected Output	REJECTED: password is too short

Test Case 8	
Purpose	Testing against not enough numbers
Input	WAYtooLong__+=8835houseing40
Previous Passwords	#558#&;DoGs GUha891)) SimPass12!!
Expected Output	REJECTED: password is too long

3 Orthogonal array testing

3.1 Introduction

4 Decision table testing

4.1 Introduction