|  |  |
| --- | --- |
| **Name:** | User, Password, Salt & Hash |
| **Summary:** | Program to store Salt & Hash generated at random for a user upon creating an account |
| **Version:** | 1.0.0 |
| **Preconditions:** | Active SQL Database preconfigured with User Tables that store Salt & Hash, Active Internet Connection for Users to create new accounts, Good Salt & Hash Functions already setup |
| **Triggers:** | User Creates or Logs into the Website |
| **Main Success Scenario:** | 1. User Requests New Account Creation via button 2. UserName & Password is requested 3. Save UserName & Password to User Class 4. Check Password Meets Standards for security purposes (If not throw error stating rule that is being broken Too Short, No Numbers, No Capitals, No Symbol) 5. Randomly generate a Salt for User Class 6. Take Password and Salt to HashGenerator to get Hash for password confirmation 7. Delete Password from our Temp location 8. Request User to verify password. 9. Take new Verify Password and sent with Salt to HashGenerator and verify against saved Hash. 10. If Fail throw error “Passwords do not match!” 11. If Success push UserName, Salt & Hash to SQL Database table “UserNames” for storage for later use. |
| **Alternative Success Scenarios:** | 1. User Requests login with UserName 2. Pull from SQL User saved data. 3. Request Password from User 4. Send Password & Salt for User to HashGenerator 5. Verify Hash Returned against saved Hash for User. 6. If Fails add to FailCounter for User save changes back to SQL & Throw error “User & Password Mismatch!” 7. If Success Reset to 0 User FailCounter and save to SQL 8. Return success to system allowing user access |
| **Postconditions:** | After system successfully creates a user SQL should have a new entry with all information pertaining to the new account. |
| **Business Rules:** | 1. System should never share with the user the Hash or Salt used for their account. Only information ever returned to the client is error messages. 2. SQL Changes should clean any inputs to avoid SQL Attacks 3. Password is over 10 Characters long 4. Password Contains 1 Upper Case Letter 5. Password Contains 1 Number 6. Password Contains 1 Symbol |
| **Notes:** | User is class Object that will save Name, Hash, Salt etc. It will never store the password itself.  HashGenerator will handle generating salts and hashing passwords.  SQL Database will ne interacted with from a SQL java class. |
| **Author:** | Collin Blake |
| **Date:** | 5-4-2019 |