Bahria University,

Karachi Campus



COURSE: CSC-221 DATA STRUCTURES AND ALGORITHM

TERM: FALL 2022, CLASS: BSE- 3 (B)

PROJECT NAME

Password Management System

Made By:

Muhammad Shoaib Akhter Qadri

Engr. Laraib Siddique/ Engr. Ayesha Khan

Signed Remarks: Score:

**TABLE OF CONTENT: Pg No**

1. INTRODUCTION & PROBLEM
2. FEATURES
3. PARADIGMS
4. ALGORITHM & EXPLANATION
5. ALGORITHM CODE
6. INTERFACES
7. CONCLUSION

* **Introduction and Problem:**

I built 5 different projects as shown below:

* **Password Management System**
* **Online Random Password Generator**
* **Encryption & Decryption Text / Password**
* **Encryption & Decryption Filing System**
* **Hashing Function (SHA-1, SHA-256, SHA-384 & SHA-512)**

If you use the same password for more than one website and might be possible that someone is hacked, you run the risk that someone will be able to use your password to break into your accounts on other sites. For that reason, you should always use a different password for every account and website you use.

Our **Password Management System** that helps you secure and organize your passwords by using the most advanced and secure **Encryption Algorithm** and **Hashing Algorithm** to encrypt your passwords information. So, no need to remember all those zillions of passwords for each and every website. Our software is an **Online Password Manager** so you can have complete control of your passwords. It also comes with a random password generator to generate unique and strong passwords.

I built also random password generator so that it generate random password with small, capital, especial symbols and numbers and spaces also. By this way, this would be easy for user to get strong password.

* **ALGORITHM & EXPLANATION:**

**Hash Algorithm:**

Hash algorithm is a cryptographic hashing algorithm used to determine the integrity of a particular piece of data. Variations of this algorithm are often used by SSL certificate authorities to sign certificates.

A common application of SHA/MD5 is to encrypting passwords, as the server side only needs to keep track of a specific user’s hash value, rather than the actual password. This is helpful in case an attacker hacks the database, as they will only find the hashed functions and not the actual passwords, so if they were to input the hashed value as a password, the hash function will convert it into another string and subsequently deny access. Additionally, SHAs exhibit the avalanche effect, where the modification of very few letters being encrypted causes a big change in output; or conversely, drastically different strings produce similar hash values. This effect causes hash values to not give any information regarding the input string, such as its original length. In addition, SHAs are also used to detect the tampering of data by attackers, where if a text file is slightly changed and barely noticeable, the modified file’s hash value will be different than the original file’s hash value, and the tampering will be rather noticeable.

**Types of Hashing:**

There are many different types of hash algorithms such as RipeMD, Tiger, xxhash and more, but the most common type of hashing are MD5, SHA-1, SHA-256, SHA-384 and SHA-512.

**1. SHA-1:**

Secure Hash Algorithm 1, or SHA-1, was developed in 1993 by the U.S. government's standards agency National Institute of Standards and Technology (NIST). It is widely used in security applications and protocols, including TLS, SSL, PGP, SSH, IPsec, and S/MIME. SHA-1 works by feeding a message as a bit string, and producing a 160-bit hash value known as a message digest.

SHA-1 is still widely used, cryptanalysts in 2005 were able to find vulnerabilities on this algorithm that detrimentally compromised its security. These vulnerabilities came in the form of an algorithm that speedily finds collisions with different inputs, meaning that two distinct inputs map to the same digest. It is more secure than MD5.

As of 2010, many organizations have recommended its replacement by SHA-2 or SHA-3. Companies like Microsoft, Google, or Mozilla have announced that their browsers will stop accepting SHA-1 encryption certificates by 2017.

**2. SHA-2:**

Due to the exposed vulnerabilities of SHA-1, cryptographers modified the algorithm to produce SHA-2, which consists of not one but two hash functions known as SHA-256 and SHA-512, using 32- and 64-bit words, respectively but we implement only SHA-256.

SHA-1 and SHA-2 differ in several ways; mainly, SHA-2 produces 224- or 256-sized digests, whereas SHA-1 produces a 160-bit digest; SHA-2 can also have block sizes that contain 1024 bits, or 512 bits, like SHA-1.

A Brute force attacks on SHA-2 are not as effective as they are against SHA-1 which makes SHA-2 a lot safer against these kinds of attacks.

**Features:**

1. Generate Random Password including upper case, lower case , numbers, and special symbols according to user-defined length.
2. Password Encryption using SHA-1, SHA-256, SHA-384 and 512 Hash.
3. Dashboard which shows all saved passwords with different website where Name , Email , username and password columns included. User can easily copy password in the clipboard.
4. User can create new record , delete record, and save record.
5. File System is use at Back-end. I save all data in firebase database.
6. User credentials to log in and verify their identities to their accounts . New user SignUp their account.
7. Password Recovery when user forget password.

**ALGORITHM CODE :**

**User Authentication:**

**Register:**

import {app,auth} from "./../firebase-config.js"

import { createUserWithEmailAndPassword,sendEmailVerification  } from "https://www.gstatic.com/firebasejs/9.12.1/firebase-auth.js";

// Sign up (Register)

let rEmail=document.getElementById("rEmail");

let rPassword=document.getElementById("rPassword");

let rCPassword=document.getElementById("rCPassword");

let writePassword=document.querySelector(".password");

let writeEmail=document.querySelector(".email")

let rSubmit=document.getElementById("rSubmit");

rSubmit.addEventListener("click",()=>{

if(rEmail.value !== ""){

if(rEmail.value.length > 8){

console.log("Successful Email")}else{

writeEmail.innerHTML='Enter correct email'

writePassword.innerHTML='';

return false}}

else{

writeEmail.innerHTML='Enter Email'

return false;}

if(rPassword.value !== ""){

if(rPassword.value.length > 8){

console.log("Successful Password")}

else{

writePassword.innerHTML='Enter strong Password'

writeEmail.innerHTML='';

return false}}

else{

writePassword.innerHTML='Enter Password'

return false;}

if(rCPassword !== ""){

if(rCPassword.value === rPassword.value){

console.log("Successful Password")}else{

writePassword.innerHTML='The password is not matched'

return false}}else{

writePassword.innerHTML='Enter Password'

return false;}

createUserWithEmailAndPassword(auth, rEmail.value, rPassword.value)

.then((userCredential) => {

const user = userCredential.user;

sendEmailVerification(auth.currentUser)

.then(() => {

location="Forget Password/Email Verification/verification.html";});

console.log(user)})

.catch((error) => {

const errorCode = error.code;

const errorMessage = error.message;

console.log(errorMessage)

swal("Sorry!", "You are not  Sign Un "+error);}); })

**Login:**

import {app,auth} from "./firebase-config.js";

import {signInWithEmailAndPassword} from 'https://www.gstatic.com/firebasejs/9.12.1/firebase-auth.js'

let lEmail=document.getElementById("lEmail");

let lPassword=document.getElementById("lPassword");

let lSubmit=document.getElementById("lSubmit");

lSubmit.addEventListener("click",()=>{

signInWithEmailAndPassword(auth, lEmail.value, lPassword.value)

.then((userCredential) => {    const user = userCredential.user;

swal("Congratulations!", "You are Sign in!", "success");

location="../passwordManagement/index.html";  })

.catch((error) => {

const errorCode = error.code;

const errorMessage = error.message;

swal("Sorry!", "You are not  sign in "+error);});})

**Online Generate Password:**

const lengthSlider = document.querySelector(".pass-length input"),

options = document.querySelectorAll(".option input"),

copyIcon = document.querySelector(".input-box span"),

passwordInput = document.querySelector(".input-box input"),

passIndicator = document.querySelector(".pass-indicator"),

generateBtn = document.querySelector(".generate-btn");

const characters = { // object of letters, numbers & symbols

lowercase: "abcdefghijklmnopqrstuvwxyz",

uppercase: "ABCDEFGHIJKLMNOPQRSTUVWXYZ",

numbers: "0123456789",

symbols: "^!$%&|[](){}:;.,\*+-#@<>~"}

const generatePassword = () => {

let staticPassword = "",

randomPassword = "",

excludeDuplicate = false,

passLength = lengthSlider.value;

options.forEach(option => { // looping through each option's checkbox

if(option.checked) { // if checkbox is checked

// if checkbox id isn't exc-duplicate && spaces

if(option.id !== "exc-duplicate" && option.id !== "spaces") {

staticPassword += characters[option.id];

} else if(option.id === "spaces") { // if checkbox id is

staticPassword += `  ${staticPassword}  `; } else {

excludeDuplicate = true;}}});

for (let i = 0; i < passLength; i++) {

let randomChar = staticPassword[Math.floor(Math.random() \* staticPassword.length)];

if(excludeDuplicate) { // if excludeDuplicate is true

!randomPassword.includes(randomChar) || randomChar == " " ? randomPassword += randomChar : i--;} else { // else add random character

randomPassword += randomChar;}}

passwordInput.value = randomPassword; // passing randomPassword to passwordInput value}

const upadatePassIndicator = () => {

passIndicator.id = lengthSlider.value <= 8 ? "weak" : lengthSlider.value <= 16 ? "medium" : "strong";}

const updateSlider = () => {

document.querySelector(".pass-length span").innerText = lengthSlider.value;

generatePassword();

upadatePassIndicator();}

updateSlider();

const copyPassword = () => {

navigator.clipboard.writeText(passwordInput.value);

copyIcon.innerText = "check"; // changing copy icon to tick

copyIcon.style.color = "#4285F4";

setTimeout(() => { // after 1500 ms, changing tick icon back to copy

copyIcon.innerText = "copy\_all";

copyIcon.style.color = "#707070";}, 1500);}

copyIcon.addEventListener("click", copyPassword);

lengthSlider.addEventListener("input", updateSlider);

generateBtn.addEventListener("click", generatePassword);

**Password Management System:**

import { initializeApp } from "https://www.gstatic.com/firebasejs/9.12.1/firebase-app.js";

import {

getFirestore,collection,query,where,onSnapshot,addDoc,getDocs,doc,updateDoc,deleteDoc,deleteField

} from "https://www.gstatic.com/firebasejs/9.12.1/firebase-firestore.js";

import {

getAuth,

onAuthStateChanged,

} from "https://www.gstatic.com/firebasejs/9.12.1/firebase-auth.js";

const firebaseConfig = {

apiKey: "AIzaSyCIVglP8b1ebS0gxo68ppitsYxdg04QXKU",

authDomain: "codingshub-f0ec1.firebaseapp.com",

databaseURL: "https://codingshub-f0ec1-default-rtdb.firebaseio.com",

projectId: "codingshub-f0ec1",

storageBucket: "codingshub-f0ec1.appspot.com",

messagingSenderId: "817602244747",

appId: "1:817602244747:web:e59247780d7e9dffd3248e",

measurementId: "G-G9P8LXN0JC"};

export const app = initializeApp(firebaseConfig);

const db = getFirestore(app);

const auth = getAuth(app);

auth.onAuthStateChanged((user) => {

if (user) {

console.log("Success");} else {

console.log("You are logout");

alert("You are logout");

location = "../userAuthentication/signup/index.html";}});

let  userName = document.querySelector("#name");

let email = document.querySelector("#email");

let websiteName = document.querySelector("#websiteName");

let password = document.querySelector("#password");

let writePassword=document.querySelector(".password")

const nameData = document.getElementById("nameData");

const emailData = document.getElementById("emailData");

const websiteNameData = document.getElementById("websiteNameData");

const passwordData = document.getElementById("passwordData");

const bodyData=document.getElementById("bodyData")

const submitDataBtn = document.querySelector("#rSubmit");

const addInfo = document.querySelector("#addInfo");

if(submitDataBtn){

submitDataBtn.addEventListener("click",()=>{

if(password.value == ""){

writePassword.innerHTML='Write your Password'

return false;}

else{

writePassword.innerHTML="";}

onAuthStateChanged (auth , async user => {

if(user){

try {

let collectionRef = collection(db, "additionalInfo")

const docRef =  addDoc(collectionRef, {

UserName: userName.value,

Email: email.value,

WebsiteName:websiteName.value,

Password:password.value,

uid: user.uid});

console.log("Document written with ID: ", docRef.id);

swal("Congratulation!", "The date is saved!", "success");} catch (e) {

console.error("Error adding document: ", e);

console.log(e)

swal("Sorry", "The data is not saved");}}

else{

console.log("User not Found");}

userName.value="";

email.value="";

websiteName.value="";

password.value="";

writePassword.innerHTML="";  });})}

var copyIcon = document.querySelector(".input-box span");

onAuthStateChanged(auth , (user) => {

if (user) {

const q = query(collection(db, "additionalInfo"), where("uid", "==", user.uid));

onSnapshot(q, (querySnapshot) => {

bodyData.innerHTML="";

querySnapshot.forEach((doc) => {

bodyData.innerHTML+=`<tr>

<td id="nameData">${doc.data().UserName}</td>

<td id="emailData">${doc.data().Email}</td>

<td id="websiteNameData">${doc.data().WebsiteName}</td>

<td id="passwordData">${doc.data().Password}</td>

<td>

<div class="input-box">

<span class="material-symbols-rounded" id="copyIcan" onclick="CopyData('${doc.data().Password}')">copy\_all</span></div></td>

<td><button type="button" class="btn btn-danger btn-sm" id="delete" onclick="deleteData('${doc.id}')">Delete</button></td></tr>`});    });}});

if(copyIcon){

copyIcon.addEventListener("click", copyPassword);}

async function CopyData(id){

navigator.clipboard.writeText(id); // copying random password}

async function deleteData(id){

const deleteRef= doc(db, "additionalInfo", id );

await deleteDoc(deleteRef,);}

if(addInfo){

addInfo.addEventListener("click",()=>{

location="addInfo.html"})}

window.deleteData = deleteData;

window.CopyData = CopyData;

**Encryption/Decryption :**

let encryptText=document.getElementById("encryptText");

let encryptedResult=document.getElementById("EncryptedResult");

let decryptText=document.getElementById("decryptText");

let decryptedResult=document.getElementById("DecryptedResult");

let encryptBtn=document.getElementById("encryptBtn");

let decryptBtn=document.getElementById("decryptBtn");

encryptBtn.addEventListener("click",()=>{

var encrypted = CryptoJS.AES.encrypt(encryptText.value,"shoaib");

encryptedResult.value = encrypted;   })

decryptBtn.addEventListener("click",()=>{

var decrypted = CryptoJS.AES.decrypt(decryptText.value,"shoaib").toString(CryptoJS.enc.Utf8);

decryptedResult.value = decrypted;})

async function fileHash(file) {

const arrayBuffer = await file.arrayBuffer();

var encrypted = CryptoJS.AES.encrypt(arrayBuffer,"shoaib");}

**Hashing Algorithm:**

let hashingText=document.getElementById("hashingText");

let hashingResult=document.getElementById("hashingResult");

let hashBtn=document.getElementById("hashBtn");

async function hash(string) {

const utf8 = new TextEncoder().encode(string);

const hashBuffer = await crypto.subtle.digest('SHA-1', utf8);

const hashArray = Array.from(new Uint8Array(hashBuffer));

const hashHex = hashArray

.map((bytes) => bytes.toString(16).padStart(2, '0'))

.join('');

return hashHex;}

hashBtn.addEventListener("click",()=>

hash(hashingText.value).then((hex) => hashingResult.value=hex))

console.log(hashingText.value)

const output = document.getElementById('FileResult');

const file = document.getElementById('input');

file.addEventListener('change', hashTheseFiles);

async function fileHash(file) {

const arrayBuffer = await file.arrayBuffer();

const hashAsArrayBuffer = await crypto.subtle.digest('SHA-1', arrayBuffer);

const uint8ViewOfHash = new Uint8Array(hashAsArrayBuffer);

const hashAsString = Array.from(uint8ViewOfHash).map((b) => b.toString(16).padStart(2, '0')).join('');

return hashAsString;}

async function hashTheseFiles(e) {

let outHTML = ''

for (const file of this.files) {

outHTML += `${await fileHash(file)}`}

output.innerHTML = outHTML;}

let fileName=document.getElementById("droppable-zone-text");

fileName.innerHTML="Drop File Here"

function getFileName(input) {

fileName.innerHTML=input.files[0].name}

**Project Interfaces:**

Graphical user interface, application

Description automatically generated**Login:**

Graphical user interface, application

Description automatically generated**Sign up:**

**Add Secret Information:**

Graphical user interface, application

Description automatically generated

**Password Management System:**

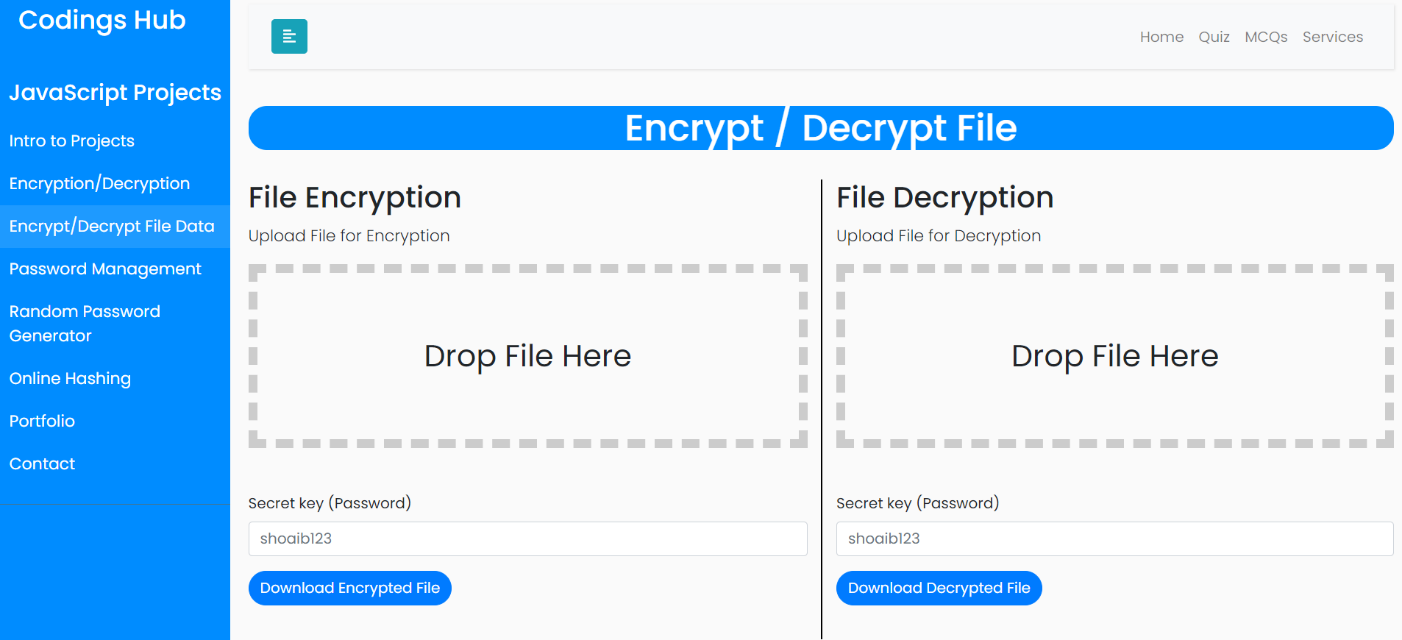
Graphical user interface

Description automatically generated with medium confidence

**Random Password Generator:**

Graphical user interface, application

Description automatically generated

**Encryption/ Decryption text:**

**Encryption/ Decryption File:**

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, application

Description automatically generated**Online Hashing Function:**

**Conclusion:**

I built these 6 different projects by different Algorithms in which Password Management System, Online Random Password Generator, Encryption / Decryption Text or Password, Encryption / Decryption File, Hashing Algorithm in which I built SHA-1, SHA-256, SHA-384 and SHA-512 as well as my Hashing Function not only hash text but also hash file system and give unique key or id of that file as you may see in pictures.

By this way, I used different Algorithm to built these perfect projects.