/\*\*

\* =============================================================================

\* @file security\_privacy\_enforcer.gs – v1.0

\* @module SECURITY\_PRIVACY\_ENFORCER

\* @description

\* وحدة مسؤولة عن تطبيق سياسات الأمان والخصوصية داخل تطبيق G-Assistant.

\* تشمل وظائف للتحقق من الأذونات، وتشفير/إخفاء البيانات الحساسة،

\* وتسجيل محاولات الوصول غير المصرح بها.

\* تعتمد على: GAssistant.Config, GAssistant.Helpers, GAssistant.AI.Dialogue

\* =============================================================================

\*/

var SECURITY\_PRIVACY\_ENFORCER = (function(GA\_Config, GA\_Helpers, GA\_AI\_Dialogue) {

'use strict';

/\*\*

\* @function checkUserPermission

\* @description

\* يتحقق مما إذا كان المستخدم الحالي لديه الإذن المطلوب لتنفيذ إجراء معين.

\* @param {string} requiredRole - الدور المطلوب لتنفيذ الإجراء (مثال: 'admin', 'editor', 'viewer').

\* @returns {boolean} - صحيح إذا كان المستخدم لديه الإذن، خطأ بخلاف ذلك.

\*/

function checkUserPermission(requiredRole) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'checkUserPermission:start', { requiredRole });

try {

const currentUserEmail = Session.getActiveUser().getEmail();

const allowedUsers = GA\_Config.getSetting('ALLOWED\_USERS') || {}; // جلب المستخدمين المسموح بهم من الإعدادات

const userRole = allowedUsers[currentUserEmail] || 'viewer'; // الافتراضي هو 'viewer'

// منطق التحقق من الدور

const rolesHierarchy = {

'admin': 3,

'editor': 2,

'viewer': 1

};

if (rolesHierarchy[userRole] >= rolesHierarchy[requiredRole]) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'checkUserPermission:granted', { userEmail: currentUserEmail, requiredRole, userRole });

return true;

} else {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'checkUserPermission:denied', { userEmail: currentUserEmail, requiredRole, userRole });

GA\_AI\_Dialogue.showError(`🚫 ليس لديك الإذن الكافي لتنفيذ هذا الإجراء. دورك الحالي: ${userRole}. الدور المطلوب: ${requiredRole}.`);

return false;

}

} catch (e) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'checkUserPermission:error', { error: e.message, requiredRole });

GA\_AI\_Dialogue.showError(`❌ حدث خطأ أثناء التحقق من الإذن: ${e.message}`);

return false;

}

}

/\*\*

\* @function encryptSensitiveData

\* @description

\* تقوم بتشفير البيانات الحساسة قبل تخزينها. (دالة وهمية للتوضيح)

\* في تطبيق حقيقي، سيتم استخدام مكتبة تشفير قوية.

\* @param {string} data - البيانات المراد تشفيرها.

\* @returns {string} - البيانات المشفرة.

\*/

function encryptSensitiveData(data) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'encryptSensitiveData:start', { dataLength: data.length });

// هذا مجرد مثال توضيحي. في تطبيق حقيقي، استخدم تشفيرًا قويًا.

const encryptedData = Utilities.base64Encode(data); // تشفير بسيط (ليس آمنًا للبيانات الحساسة جدًا)

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'encryptSensitiveData:success');

return encryptedData;

}

/\*\*

\* @function decryptSensitiveData

\* @description

\* تقوم بفك تشفير البيانات الحساسة بعد استرجاعها. (دالة وهمية للتوضيح)

\* @param {string} encryptedData - البيانات المشفرة.

\* @returns {string} - البيانات الأصلية.

\*/

function decryptSensitiveData(encryptedData) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'decryptSensitiveData:start', { dataLength: encryptedData.length });

try {

const decryptedData = Utilities.newBlob(Utilities.base64Decode(encryptedData)).getDataAsString();

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'decryptSensitiveData:success');

return decryptedData;

} catch (e) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'decryptSensitiveData:error', { error: e.message });

GA\_AI\_Dialogue.showError(`❌ فشل فك تشفير البيانات: ${e.message}`);

return null;

}

}

/\*\*

\* @function enforceDataMasking

\* @description

\* تقوم بإخفاء أجزاء من البيانات الحساسة (مثل أرقام بطاقات الائتمان أو هويات المستخدمين) للعرض.

\* @param {string} data - البيانات الأصلية.

\* @param {string} type - نوع البيانات (مثال: 'credit\_card', 'email').

\* @returns {string} - البيانات المخفية.

\*/

function enforceDataMasking(data, type) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'enforceDataMasking:start', { type, dataLength: data.length });

let maskedData = data;

switch (type) {

case 'credit\_card':

maskedData = data.replace(/(\d{12})(\d{4})/, '\*\*\*\*\*\*\*\*\*\*\*\*$2'); // إخفاء أول 12 رقمًا

break;

case 'email':

maskedData = data.replace(/(?<=.{3}).(?=.\*@)/g, '\*'); // إخفاء جزء من البريد الإلكتروني

break;

// يمكن إضافة المزيد من أنواع الإخفاء هنا

}

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', 'enforceDataMasking:success', { type });

return maskedData;

}

/\*\*

\* @function logSecurityEvent

\* @description

\* تسجل حدثًا أمنيًا (مثل محاولة وصول غير مصرح بها) في سجل الأمان.

\* @param {string} eventType - نوع الحدث الأمني (مثال: 'unauthorized\_access', 'data\_tampering').

\* @param {object} details - تفاصيل إضافية حول الحدث.

\*/

function logSecurityEvent(eventType, details) {

GA\_Helpers.logOperation('SECURITY\_PRIVACY\_ENFORCER', `securityEvent:${eventType}`, details, 'security\_log'); // استخدام 'security\_log' كنوع سجل مخصص

// يمكن هنا إضافة منطق لإرسال تنبيهات للمسؤولين أو حفظ السجل في مكان آمن

console.warn(`SECURITY ALERT: Event Type: ${eventType}, Details: ${JSON.stringify(details)}`);

}

// --- تصدير الواجهة العامة للوحدة ---

return {

checkUserPermission,

encryptSensitiveData,

decryptSensitiveData,

enforceDataMasking,

logSecurityEvent

};

})(

GAssistant.Config,

GAssistant.Helpers,

GAssistant.AI.Dialogue

);