/\*\*

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

\* @file tests.gs – وحدة اختبارات ذكية للمساعد الذكي (v2.5)

\* @description

\* يحتوي على اختبارات تكاملية لجميع الموديولات المهمة في النظام.

\* تم الفصل بين التنفيذ والعرض لدعم التكامل مع الشريط الجانبي.

\* تمت إضافة اختبارات لوظيفة البحث عن دوال Apps Script المدمجة، ووظائف توثيق الأدوات الداخلية.

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

\* GAssistant.Tools.Sheets, GAssistant.Tools.Accounting, GAssistant.Tools.General,

\* GAssistant.BuiltinDocs, GAssistant.Tools.DocGen

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

\*/

var GAssistant = GAssistant || {};

// حقن التبعيات: يجب تمرير جميع الوحدات المعتمدة هنا بأسماء مساحات الأسماء الكاملة

GAssistant.Tests = (function(

GA\_Helpers,

GA\_AI\_Dialogue,

GA\_AI\_Core,

GA\_Tools\_Sheets,

GA\_Tools\_Accounting,

GA\_Tools\_General,

GA\_BuiltinDocs,

GA\_Tools\_DocGen

) {

'use strict';

/\*\*

\* @function getAllTestResults

\* @description يُعيد نتائج كل الاختبارات على شكل مصفوفة جاهزة للعرض.

\* @returns {Array<{name:string, success:boolean, message:string}>} - مصفوفة كائنات نتائج الاختبار.

\*/

function getAllTestResults() {

GA\_Helpers.logOperation('GAssistant.Tests', 'getAllTestResults:start', {});

const results = [

testModuleExists("GAssistant.Tools.General"),

testModuleExists("GAssistant.Tools.Sheets"),

testModuleExists("GAssistant.Tools.Accounting"),

testModuleExists("GAssistant.AI.Core"),

testModuleExists("GAssistant.BuiltinDocs"),

testModuleExists("GAssistant.Tools.DocGen"),

testAskCoreWorks(),

testHighlightEmptyCells(),

testAccountingLogRevenue(),

testAccountingLogExpense(),

testWorkshopProposal(),

testBuiltinFunctionDocSearch(),

testGenerateToolMetadata(),

testExportToolsDocumentationToDoc()

];

GA\_Helpers.logOperation('GAssistant.Tests', 'getAllTestResults:end', { count: results.length });

return results;

}

/\*\*

\* @function runAllTests

\* @description تشغيل كل الاختبارات وعرضها كـ Alert تقليدي.

\* @returns {void}

\*/

function runAllTests() {

GA\_Helpers.logOperation('GAssistant.Tests', 'runAllTests:start', {});

const results = getAllTestResults();

const passed = results.filter(r => r.success).length;

const total = results.length;

const summary = results.map(r => {

const icon = r.success ? '✅' : '❌';

const name = r.name.padEnd(30);

return `${icon} ${name} → ${r.message}`;

});

const timestamp = Utilities.formatDate(new Date(), Session.getScriptTimeZone(), "yyyy-MM-dd HH:mm");

const message = `🧪 نتائج الاختبارات (${passed}/${total}) – ${timestamp}\n\n` + summary.join('\n');

GA\_AI\_Dialogue.showMessage('اختبارات النظام', message);

GA\_Helpers.logOperation('GAssistant.Tests', 'runAllTests:end', { passed, total });

}

/\*\*

\* @function testModuleExists

\* @description يتحقق من وجود وحدة بالاسم المحدد.

\* @param {string} name - اسم الوحدة المراد التحقق منها (مثال: "GAssistant.Tools.General").

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testModuleExists(name) {

try {

// استخدام eval للتحقق من وجود الوحدة بناءً على اسمها الكامل

const exists = typeof eval(name) !== 'undefined';

GA\_Helpers.logOperation('GAssistant.Tests', `testModuleExists:${name}`, { exists });

return {

name: `Module: ${name}`,

success: exists,

message: exists ? 'موجود' : '❌ غير موجود'

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', `testModuleExists:${name}:error`, { error: e.message });

return { name: `Module: ${name}`, success: false, message: e.message };

}

}

/\*\*

\* @function testAskCoreWorks

\* @description اختبار GAssistant.AI.Core.ask للتأكد من أنها ترجع محتوى.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testAskCoreWorks() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testAskCoreWorks:start', {});

try {

const response = GA\_AI\_Core.ask("ما هي وظيفتك؟");

const content = response?.content || '';

const success = Boolean(content);

GA\_Helpers.logOperation('GAssistant.Tests', 'testAskCoreWorks:end', { success, contentPreview: content.slice(0, 40) });

return {

name: "GAssistant.AI.Core.ask",

success,

message: success ? content.slice(0, 40) + '...' : 'لا محتوى مرجع'

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testAskCoreWorks:error', { error: e.message });

return { name: "GAssistant.AI.Core.ask", success: false, message: e.message };

}

}

/\*\*

\* @function testHighlightEmptyCells

\* @description اختبار تمييز الخلايا الفارغة باستخدام GAssistant.Tools.Sheets.highlightEmptyCells.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testHighlightEmptyCells() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testHighlightEmptyCells:start', {});

const sheetName = '🚫 خلايا فارغة اختبار'; // اسم ورقة فريد للاختبار

try {

const ss = SpreadsheetApp.getActiveSpreadsheet();

let sheet = ss.getSheetByName(sheetName);

if (sheet) ss.deleteSheet(sheet); // حذف الورقة إذا كانت موجودة من اختبار سابق

sheet = ss.insertSheet(sheetName);

sheet.getRange("A1:B2").setValues([["", "نص"], ["", ""]]);

const msg = GA\_Tools\_Sheets.highlightEmptyCells(sheetName, "A1:B2", "#ffeeee");

const success = msg?.type === 'success';

GA\_Helpers.logOperation('GAssistant.Tests', 'testHighlightEmptyCells:end', { success, message: msg?.content });

return {

name: "GAssistant.Tools.Sheets.highlightEmptyCells",

success: success,

message: msg?.content || 'تم التمييز'

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testHighlightEmptyCells:error', { error: e.message });

return { name: "GAssistant.Tools.Sheets.highlightEmptyCells", success: false, message: e.message };

} finally {

const ss = SpreadsheetApp.getActiveSpreadsheet();

const sheet = ss.getSheetByName(sheetName);

if (sheet) ss.deleteSheet(sheet);

}

}

/\*\*

\* @function testAccountingLogRevenue

\* @description اختبار إدخال إيرادات باستخدام GAssistant.Tools.Accounting.logRevenueEntry.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testAccountingLogRevenue() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testAccountingLogRevenue:start', {});

try {

const res = GA\_Tools\_Accounting.logRevenueEntry({

date: Utilities.formatDate(new Date(), Session.getScriptTimeZone(), "yyyy-MM-dd"),

product: "منتج اختبار",

qty: 1,

unitPrice: 100

});

const success = res?.type === 'success';

GA\_Helpers.logOperation('GAssistant.Tests', 'testAccountingLogRevenue:end', { success, message: res?.content });

return {

name: "GAssistant.Tools.Accounting.logRevenueEntry",

success: success,

message: res?.content || 'تم التسجيل'

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testAccountingLogRevenue:error', { error: e.message });

return { name: "GAssistant.Tools.Accounting.logRevenueEntry", success: false, message: e.message };

}

}

/\*\*

\* @function testAccountingLogExpense

\* @description اختبار إدخال مصروف باستخدام GAssistant.Tools.Accounting.logExpenseEntry.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testAccountingLogExpense() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testAccountingLogExpense:start', {});

try {

const res = GA\_Tools\_Accounting.logExpenseEntry({

date: Utilities.formatDate(new Date(), Session.getScriptTimeZone(), "yyyy-MM-dd"),

description: "فحص كهرباء",

amount: 75,

category: "خدمات"

});

const success = res?.type === 'success';

GA\_Helpers.logOperation('GAssistant.Tests', 'testAccountingLogExpense:end', { success, message: res?.content });

return {

name: "GAssistant.Tools.Accounting.logExpenseEntry",

success: success,

message: res?.content || 'تم التسجيل'

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testAccountingLogExpense:error', { error: e.message });

return { name: "GAssistant.Tools.Accounting.logExpenseEntry", success: false, message: e.message };

}

}

/\*\*

\* @function testWorkshopProposal

\* @description اختبار اقتراح كود إلى ورشة العمل باستخدام GAssistant.Tools.General.proposeAndStageCode.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testWorkshopProposal() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testWorkshopProposal:start', {});

try {

const fnName = "اختبار\_ورشة";

const description = "وصف اختبار اقتراح كود";

const fnCode = "function اختبار\_ورشة() { Logger.log('ok'); }";

const result = GA\_Tools\_General.proposeAndStageCode(fnName, description, fnCode);

const success = result?.type === 'success';

GA\_Helpers.logOperation('GAssistant.Tests', 'testWorkshopProposal:end', { success, message: result?.content });

return {

name: "GAssistant.Tools.General.proposeAndStageCode",

success: success,

message: result?.content || 'تمت إضافة الاقتراح'

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testWorkshopProposal:error', { error: e.message });

return { name: "GAssistant.Tools.General.proposeAndStageCode", success: false, message: e.message };

}

}

/\*\*

\* @function testBuiltinFunctionDocSearch

\* @description اختبار وظيفة البحث عن دوال Apps Script المدمجة باستخدام GAssistant.BuiltinDocs.getBuiltinFunctionDoc.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testBuiltinFunctionDocSearch() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testBuiltinFunctionDocSearch:start', {});

try {

const knownFunctionResult = GA\_BuiltinDocs.getBuiltinFunctionDoc("SpreadsheetApp.getActiveSpreadsheet");

const knownFunctionSuccess = knownFunctionResult && (knownFunctionResult.table || knownFunctionResult.text) &&

(knownFunctionResult.text.includes("SpreadsheetApp.getActiveSpreadsheet") || knownFunctionResult.text.includes("Returns the currently active spreadsheet"));

const unknownFunctionResult = GA\_BuiltinDocs.getBuiltinFunctionDoc("NonExistentAppsScriptFunction123");

const unknownFunctionSuccess = unknownFunctionResult && unknownFunctionResult.text &&

unknownFunctionResult.text.includes("تعذر الحصول على توثيق لهذه الدالة");

const success = knownFunctionSuccess && unknownFunctionSuccess;

let message = "";

if (knownFunctionSuccess) {

message += "✅ بحث عن دالة معروفة نجح. ";

} else {

message += "❌ بحث عن دالة معروفة فشل. ";

}

if (unknownFunctionSuccess) {

message += "✅ بحث عن دالة غير موجودة نجح.";

} else {

message += "❌ بحث عن دالة غير موجودة فشل.";

}

GA\_Helpers.logOperation('GAssistant.Tests', 'testBuiltinFunctionDocSearch:end', { success, message });

return {

name: "GAssistant.BuiltinDocs.getBuiltinFunctionDoc",

success: success,

message: message

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testBuiltinFunctionDocSearch:error', { error: e.message });

return { name: "GAssistant.BuiltinDocs.getBuiltinFunctionDoc", success: false, message: e.message };

}

}

/\*\*

\* @function testGenerateToolMetadata

\* @description اختبار وظيفة توليد البيانات الوصفية للأدوات من GAssistant.Tools.DocGen.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testGenerateToolMetadata() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testGenerateToolMetadata:start', {});

try {

const result = GA\_Tools\_DocGen.generateToolMetadata();

const success = result && typeof result === 'string' && result.includes('تم تحليل وتوثيق') && result.includes('أداة بنجاح');

GA\_Helpers.logOperation('GAssistant.Tests', 'testGenerateToolMetadata:end', { success, message: result });

return {

name: "GAssistant.Tools.DocGen.generateToolMetadata",

success: success,

message: success ? 'تم توليد البيانات الوصفية بنجاح.' : `فشل توليد البيانات الوصفية: ${result}`

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testGenerateToolMetadata:error', { error: e.message });

return { name: "GAssistant.Tools.DocGen.generateToolMetadata", success: false, message: e.message };

}

}

/\*\*

\* @function testExportToolsDocumentationToDoc

\* @description اختبار وظيفة تصدير توثيق الأدوات إلى مستند Google Docs من GAssistant.Tools.DocGen.

\* @returns {object} - كائن نتيجة الاختبار.

\*/

function testExportToolsDocumentationToDoc() {

GA\_Helpers.logOperation('GAssistant.Tests', 'testExportToolsDocumentationToDoc:start', {});

try {

const result = GA\_Tools\_DocGen.exportToolsDocumentationToDoc();

const success = result && typeof result === 'string' && result.includes('تم إنشاء المستند:');

GA\_Helpers.logOperation('GAssistant.Tests', 'testExportToolsDocumentationToDoc:end', { success, message: result });

return {

name: "GAssistant.Tools.DocGen.exportToolsDocumentationToDoc",

success: success,

message: success ? 'تم تصدير المستند بنجاح.' : `فشل تصدير المستند: ${result}`

};

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tests', 'testExportToolsDocumentationToDoc:error', { error: e.message });

return { name: "GAssistant.Tools.DocGen.exportToolsDocumentationToDoc", success: false, message: e.message };

}

}

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

return {

getAllTestResults,

runAllTests

};

})(

GAssistant.Helpers,

GAssistant.AI.Dialogue,

GAssistant.AI.Core,

GAssistant.Tools.Sheets,

GAssistant.Tools.Accounting,

GAssistant.Tools.General,

GAssistant.BuiltinDocs,

GAssistant.Tools.DocGen

);

// ---------------------------------------------------------------------------

// الدوال العامة (Global Functions)

// ---------------------------------------------------------------------------

// هذه الدوال ضرورية لتمكين google.script.run من استدعاء الدوال من HTML

// ولتوفير نقاط دخول للقوائم المخصصة في Google Sheets.

// يجب أن تستدعي هذه الدوال الواجهة العامة لـ GAssistant.Tests.

/\*\*

\* @function onOpen

\* @description دالة نقطة الدخول القياسية لـ Google Apps Script التي تُستدعى عند فتح المصنف.

\* تُنشئ قائمة "Test Suite" في واجهة المستخدم.

\* @param {GoogleAppsScript.Events.SheetsOnOpen} [e] - كائن حدث الفتح (اختياري).

\*/

function onOpen(e) {

// استخدام الوحدة GAssistant.Tests المصدرة

if (typeof GAssistant.Tests !== 'undefined' && GAssistant.Tests.runAllTests) {

SpreadsheetApp.getUi()

.createMenu("🧪 Test Suite")

.addItem("✅ تشغيل Alert", "GAssistant.Tests.runAllTests") // استدعاء من الوحدة

.addItem("📊 نتائج في Sidebar", "showTestResultsSidebar") // هذه الدالة لا تزال عامة

.addToUi();

} else {

SpreadsheetApp.getUi()

.createMenu("🧪 Test Suite (Error)")

.addItem("❌ لا يمكن تحميل الاختبارات", "showErrorLoadingTests")

.addToUi();

}

}

/\*\*

\* @function showTestResultsSidebar

\* @description تُظهر الشريط الجانبي لعرض نتائج الاختبارات.

\* @returns {void}

\*/

function showTestResultsSidebar() {

const html = HtmlService.createHtmlOutputFromFile("TestResultsSidebar")

.setTitle("🧪 نتائج الاختبارات")

.setWidth(420);

SpreadsheetApp.getUi().showSidebar(html);

}

/\*\*

\* @function showErrorLoadingTests

\* @description دالة مساعدة لعرض رسالة خطأ إذا فشلت الاختبارات في التحميل.

\* @returns {void}

\*/

function showErrorLoadingTests() {

// افتراض وجود GAssistant.AI.Dialogue

if (typeof GAssistant.AI.Dialogue !== 'undefined' && GAssistant.AI.Dialogue.showError) {

GAssistant.AI.Dialogue.showError('فشل تحميل وحدة الاختبارات. يرجى التحقق من ملف tests.gs.');

} else {

SpreadsheetApp.getUi().alert('فشل تحميل وحدة الاختبارات. يرجى التحقق من ملف tests.gs.');

}

}

/\*\*

\* @function getAllTestResults

\* @description دالة عامة تُستخدم بواسطة google.script.run لجلب جميع نتائج الاختبارات.

\* @returns {Array<{name:string, success:boolean, message:string}>} - مصفوفة كائنات نتائج الاختبار.

\*/

function getAllTestResults() {

// التأكد من أن الوحدة GAssistant.Tests موجودة قبل استدعاء دالتها

if (typeof GAssistant.Tests !== 'undefined' && GAssistant.Tests.getAllTestResults) {

return GAssistant.Tests.getAllTestResults();

} else {

// في حالة عدم تحميل الوحدة، إرجاع رسالة خطأ

return [{

name: "Error Loading Tests",

success: false,

message: "وحدة الاختبارات (GAssistant.Tests) غير متاحة."

}];

}

}