/\*\*

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

\* @file gemini\_script\_doctor.gs – v1.0

\* @module GAssistant.AI.ScriptDoctor

\* @description

\* وحدة طبيب السكربت الذكي. تستخدم نموذج Gemini لتحليل الأكواد البرمجية،

\* وتقديم تغذية راجعة، اقتراحات لتحسين الأداء والقراءة، وتحديد المشاكل المحتملة.

\* يمكن توسيعها لاحقًا لتوليد اختبارات الوحدات وفحص الأمان.

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

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

\*/

var GAssistant = GAssistant || {};

GAssistant.AI = GAssistant.AI || {};

GAssistant.AI.ScriptDoctor = (function(

GA\_Helpers,

GA\_AI\_Dialogue,

GA\_AI\_Core,

GA\_SettingsProvider

) {

'use strict';

// API Key for Gemini API. This will be automatically provided by the Canvas environment.

// Note: GA\_AI\_Core should handle the API key internally, but it's good practice

// to ensure it's available if needed for direct API calls.

const API\_KEY = ""; // This is typically managed by AI\_CORE or a dedicated config module.

/\*\*

\* @typedef {Object} CodeReviewFeedback

\* @property {string} summary - ملخص عام للمراجعة.

\* @property {Array<Object>} feedback - مصفوفة من كائنات التغذية الراجعة.

\* @property {string} feedback[].category - فئة الملاحظة (مثال: 'Performance', 'Readability', 'Security').

\* @property {string} feedback[].title - عنوان موجز للملاحظة.

\* @property {string} feedback[].description - وصف تفصيلي للملاحظة.

\* @property {string} [feedback[].codeSnippet] - جزء من الكود ذو الصلة بالملاحظة.

\* @property {string} [patch] - اقتراح لتعديل الكود بصيغة Unified Diff.

\*/

/\*\*

\* @function reviewCode

\* @description

\* ترسل الكود إلى نموذج Gemini لمراجعته بناءً على نوع المراجعة المطلوب.

\* @param {string} code - الكود البرمجي المراد مراجعته.

\* @param {'general'|'refactor'|'security'|'unit\_tests'} reviewType - نوع المراجعة المطلوبة.

\* @returns {CodeReviewFeedback} - كائن يحتوي على ملخص المراجعة، الملاحظات، و patch إذا تم توليده.

\*/

function reviewCode(code, reviewType = 'general') {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'reviewCode:start', { length: code.length, reviewType: reviewType });

const settings = GA\_SettingsProvider.getSettings();

const model = settings.CODE\_REVIEW\_MODEL || "gemini-1.5-pro"; // استخدام نموذج أقوى لمراجعة الكود

let prompt = `أنت خبير في Google Apps Script ومراجعة الكود. قم بمراجعة الكود التالي بناءً على نوع المراجعة "${reviewType}".

قدم تقييمًا عامًا (overallScore من 0 إلى 100)، وملخصًا، وقائمة مفصلة بالملاحظات (feedbackItems).

لكل ملاحظة، حدد الفئة (Category)، العنوان (Title)، الوصف (Description)، وأي جزء من الكود ذو صلة (codeSnippet).

إذا كانت المراجعة من نوع 'refactor' أو 'general' وتوجد تحسينات واضحة، قم بتوليد اقتراح لتعديل الكود (patch) بصيغة Unified Diff.

يجب أن يكون الرد بتنسيق JSON، مع التأكد من أن جميع المفاتيح موجودة حتى لو كانت القيم فارغة.

الكود للمراجعة:

\`\`\`javascript

${code}

\`\`\`

مثال على هيكل JSON المطلوب:

\`\`\`json

{

"overallScore": 85,

"summary": "ملخص عام للمراجعة.",

"feedbackItems": [

{

"category": "Readability",

"title": "تحسين أسماء المتغيرات",

"description": "استخدم أسماء متغيرات أكثر وصفية لتحسين قابلية القراءة.",

"codeSnippet": "var x = 10;"

},

{

"category": "Performance",

"title": "تجنب العمليات المتكررة على SpreadsheetApp",

"description": "استدعاء SpreadsheetApp.getActiveSpreadsheet() داخل حلقة يقلل الأداء.",

"codeSnippet": "for (...) { SpreadsheetApp.getActiveSpreadsheet(); }"

}

],

"patch": "--- a/old\_file.js\\n+++ b/new\_file.js\\n@@ -1,3 +1,4 @@\\n function oldFunc() {\\n- console.log('old');\\n+ console.log('new');\\n }\\n"

}

\`\`\`

`;

const payload = {

contents: [{ role: "user", parts: [{ text: prompt }] }],

generationConfig: {

responseMimeType: "application/json",

responseSchema: {

type: "OBJECT",

properties: {

"overallScore": { "type": "NUMBER" },

"summary": { "type": "STRING" },

"feedbackItems": {

"type": "ARRAY",

"items": {

"type": "OBJECT",

"properties": {

"category": { "type": "STRING" },

"title": { "type": "STRING" },

"description": { "type": "STRING" },

"codeSnippet": { "type": "STRING" }

}

}

},

"patch": { "type": "STRING" }

}

}

}

};

try {

// استخدام GA\_AI\_Core.ask لإرسال الطلب

const result = GA\_AI\_Core.ask(payload, model);

if (result && result.type === 'structured\_data' && result.content) {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'reviewCode:success', { score: result.content.overallScore });

return result.content; // إرجاع الكائن JSON المحلل

} else if (result && result.type === 'message' && result.level === 'error') {

// إذا كان هناك خطأ من AI\_Core

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'reviewCode:ai\_core\_error', { error: result.text });

return {

overallScore: 0,

summary: `خطأ من نموذج Gemini: ${result.text}`,

feedbackItems: [],

patch: null,

error: result.text

};

} else {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'reviewCode:unexpected\_response', { result });

throw new Error("استجابة غير متوقعة من نموذج Gemini.");

}

} catch (e) {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'reviewCode:failure', { error: e.message });

return {

overallScore: 0,

summary: `حدث خطأ أثناء مراجعة الكود: ${e.message}`,

feedbackItems: [],

patch: null,

error: e.message

};

}

}

/\*\*

\* @function getReviewFeedbackForSidebar

\* @description

\* دالة وسيطة لـ reviewCode مصممة خصيصًا للاستخدام من الشريط الجانبي (UI).

\* تقوم بمعالجة النتيجة لتكون جاهزة للعرض في HTML.

\* @param {string} code - الكود البرمجي المراد مراجعته.

\* @param {string} reviewType - نوع المراجعة المطلوبة.

\* @returns {object} - كائن جاهز للعرض في الشريط الجانبي.

\*/

function getReviewFeedbackForSidebar(code, reviewType) {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'getReviewFeedbackForSidebar:start', { reviewType: reviewType });

try {

const result = reviewCode(code, reviewType);

// هنا يمكن إضافة منطق إضافي لتنسيق النتيجة لـ HTML إذا لزم الأمر

// حاليًا، نفترض أن الواجهة الأمامية ستتعامل مع هيكل JSON مباشرة.

return result;

} catch (e) {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'getReviewFeedbackForSidebar:error', { error: e.message });

GA\_AI\_Dialogue.showError(`❌ فشل في جلب مراجعة الكود: ${e.message}`);

return { overallScore: 0, summary: `🚨 حدث خطأ في الخادم: ${e.message}`, feedbackItems: [], patch: null, error: e.message };

}

}

/\*\*

\* @function generateUnitTestSuggestions

\* @description

\* يولد اقتراحات لاختبارات الوحدة لكود معين.

\* (هذه دالة هيكلية، يمكن تطويرها لاستدعاء Gemini خصيصًا لتوليد الاختبارات).

\* @param {string} code - الكود المراد توليد اختبارات له.

\* @returns {object} - كائن يحتوي على اقتراحات اختبارات الوحدة.

\*/

function generateUnitTestSuggestions(code) {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'generateUnitTestSuggestions:start', { codePreview: code.substring(0, 50) + '...' });

// هنا يمكن استدعاء reviewCode مع reviewType 'unit\_tests'

const result = reviewCode(code, 'unit\_tests');

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'generateUnitTestSuggestions:end', { success: Boolean(result.feedback.length) });

return result;

}

/\*\*

\* @function performSecurityScan

\* @description

\* يقوم بفحص أمان لكود معين.

\* (هذه دالة هيكلية، يمكن تطويرها لاستدعاء Gemini خصيصًا لفحص الأمان).

\* @param {string} code - الكود المراد فحصه.

\* @returns {object} - كائن يحتوي على نتائج فحص الأمان.

\*/

function performSecurityScan(code) {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'performSecurityScan:start', { codePreview: code.substring(0, 50) + '...' });

// هنا يمكن استدعاء reviewCode مع reviewType 'security'

// أو التفاعل مباشرة مع وحدة tools\_security\_review.gs عندما تكون جاهزة.

const result = reviewCode(code, 'security');

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'performSecurityScan:end', { success: Boolean(result.feedback.length) });

return result;

}

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

return {

reviewCode: reviewCode,

getReviewFeedbackForSidebar: getReviewFeedbackForSidebar,

generateUnitTestSuggestions: generateUnitTestSuggestions,

performSecurityScan: performSecurityScan

};

})(

GAssistant.Helpers,

GAssistant.AI.Dialogue,

GAssistant.AI.Core,

GAssistant.Config // تأكد من تمرير GAssistant.Config هنا

);

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

// الدوال العامة (Global Functions)

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

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

// ولتوفير نقاط دخول للقوائم المخصصة في Google Sheets.

/\*\*

\* @function getReviewFeedbackForSidebar

\* @description

\* دالة عامة تُستخدم بواسطة google.script.run لجلب تغذية راجعة حول الكود لعرضها في الشريط الجانبي.

\* @param {string} code - الكود البرمجي المراد مراجعته.

\* @param {string} reviewType - نوع المراجعة المطلوبة.

\* @returns {object} - كائن جاهز للعرض في الشريط الجانبي.

\*/

function getReviewFeedbackForSidebar(code, reviewType) {

return GAssistant.AI.ScriptDoctor.getReviewFeedbackForSidebar(code, reviewType);

}