/\*\*

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

\* @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 - تصنيف الاقتراح (مثال: 'readability', 'performance', 'refactoring').

\* @property {string} feedback[].message - وصف المشكلة أو الاقتراح.

\* @property {string} [feedback[].suggestion] - اقتراح كود بديل أو تفصيلي.

\*/

/\*\*

\* @function reviewCode

\* @description

\* 🧠 يحلل الكود البرمجي باستخدام نموذج Gemini ويقدم تغذية راجعة.

\* يمكن تحديد نوع المراجعة (عام، إعادة هيكلة، أمان، اختبارات).

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

\* @param {string} reviewType - نوع المراجعة المطلوبة ('general', 'refactor', 'security', 'unit\_tests').

\* @returns {CodeReviewFeedback} - كائن يحتوي على ملخص ومصفوفة من التغذية الراجعة.

\*/

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

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', `reviewCode:${reviewType}:start`, { codePreview: code.substring(0, 100) + '...' });

try {

if (!code || typeof code !== 'string') {

throw new Error("الكود المراد مراجعته غير صالح.");

}

const prompt = \_buildReviewPrompt(code, reviewType);

// استخدام GA\_AI\_Core.ask للتفاعل مع نموذج Gemini

const response = GA\_AI\_Core.ask(prompt, {

responseMimeType: "application/json",

responseSchema: {

type: "OBJECT",

properties: {

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

"feedback": {

"type": "ARRAY",

"items": {

"type": "OBJECT",

"properties": {

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

"message": { "type": "STRING" },

"suggestion": { "type": "STRING" }

},

"propertyOrdering": ["category", "message", "suggestion"]

}

}

},

"propertyOrdering": ["summary", "feedback"]

}

});

const reviewResult = \_parseGeminiResponse(response);

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', `reviewCode:${reviewType}:success`, { summary: reviewResult.summary.substring(0, 50) + '...' });

return reviewResult;

} catch (e) {

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

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

return {

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

feedback: [{ category: 'error', message: `فشل مراجعة الكود: ${e.message}` }]

};

}

}

/\*\*

\* @function \_buildReviewPrompt

\* @description

\* 📝 يبني الموجه (prompt) لنموذج Gemini بناءً على نوع المراجعة المطلوبة.

\* @private

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

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

\* @returns {string} - الموجه الكامل لـ Gemini.

\*/

function \_buildReviewPrompt(code, reviewType) {

let instruction = "";

switch (reviewType) {

case 'refactor':

instruction = "قم بمراجعة الكود التالي واقترح تحسينات لإعادة الهيكلة، مع التركيز على الوضوح، قابلية الصيانة، وتقليل التكرار. قدم اقتراحات محددة للكود البديل.";

break;

case 'security':

instruction = "قم بمراجعة الكود التالي بحثًا عن أي ثغرات أمنية محتملة أو ممارسات غير آمنة في Google Apps Script. اذكر المخاوف الأمنية وقدم توصيات للحل.";

break;

case 'unit\_tests':

instruction = "قم بتحليل الكود التالي واقترح أفكارًا لاختبارات الوحدات (Unit Tests) التي تغطي حالات الاستخدام الرئيسية والحالات الهامشية (edge cases).";

break;

case 'general':

default:

instruction = "قم بمراجعة الكود التالي وقدم تغذية راجعة عامة حول جودته، قابليته للقراءة، أدائه، وأي مشاكل محتملة. اقترح تحسينات حيثما أمكن.";

break;

}

return `${instruction}\n\nالكود:\n\`\`\`javascript\n${code}\n\`\`\`\n\nقدم الرد بصيغة JSON تحتوي على 'summary' (ملخص عام) ومصفوفة 'feedback' لكل اقتراح. كل عنصر في 'feedback' يجب أن يحتوي على 'category' (مثل 'readability', 'performance', 'security', 'refactoring', 'unit\_test\_idea'), و 'message' (وصف المشكلة/الاقتراح), و 'suggestion' (كود مقترح أو تفاصيل إضافية).`;

}

/\*\*

\* @function \_parseGeminiResponse

\* @description

\* 🔄 يحلل استجابة نموذج Gemini JSON ويستخرج منها التغذية الراجعة.

\* @private

\* @param {object} geminiResponse - الكائن الكامل للاستجابة من GA\_AI\_Core.ask.

\* @returns {CodeReviewFeedback} - كائن التغذية الراجعة المهيكل.

\*/

function \_parseGeminiResponse(geminiResponse) {

if (geminiResponse && geminiResponse.content) {

try {

const parsedContent = JSON.parse(geminiResponse.content);

// التحقق من صحة البنية المتوقعة

if (parsedContent.summary && Array.isArray(parsedContent.feedback)) {

return parsedContent;

} else {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'parseGeminiResponse:invalid\_structure', { response: geminiResponse.content });

return {

summary: "فشل تحليل استجابة النموذج: بنية غير متوقعة.",

feedback: [{ category: 'error', message: "فشل تحليل استجابة النموذج: بنية غير متوقعة." }]

};

}

} catch (e) {

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'parseGeminiResponse:json\_parse\_error', { error: e.message, content: geminiResponse.content });

return {

summary: `فشل تحليل استجابة النموذج (JSON): ${e.message}`,

feedback: [{ category: 'error', message: `فشل تحليل استجابة النموذج (JSON): ${e.message}` }]

};

}

}

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', 'parseGeminiResponse:empty\_response', {});

return {

summary: "لم يتم استلام محتوى من النموذج.",

feedback: [{ category: 'info', message: "لم يتم استلام محتوى من النموذج." }]

};

}

/\*\*

\* @function getReviewFeedbackForSidebar

\* @description

\* 📊 واجهة عامة تستخدمها الواجهة الأمامية (Sidebar) لجلب تغذية راجعة حول الكود.

\* تقوم باستدعاء `reviewCode` وتنسيق النتائج للعرض 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) {

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

try {

const reviewResult = reviewCode(code, reviewType);

// هنا يمكننا تحويل الـ JSON إلى تنسيق HTML أو نص منسق

// ليتم عرضه مباشرة في Sidebar.

// هذا مثال بسيط، ويمكن توسيعه لإنشاء بطاقات أو جداول أكثر تعقيدًا.

let htmlOutput = `<h3 style="color:#004085; text-align:center; margin-bottom:15px;">ملخص المراجعة:</h3>`;

htmlOutput += `<p style="background-color:#e6f7ff; padding:10px; border-radius:5px; margin-bottom:20px; text-align:right;">${reviewResult.summary}</p>`;

if (reviewResult.feedback && reviewResult.feedback.length > 0) {

htmlOutput += `<h4 style="color:#0056b3; margin-bottom:10px; text-align:right;">التغذية الراجعة التفصيلية:</h4>`;

reviewResult.feedback.forEach(item => {

htmlOutput += `

<div style="background-color:#fff; border:1px solid #ddd; border-radius:8px; padding:12px; margin-bottom:10px; box-shadow: 0 2px 4px rgba(0,0,0,0.05);">

<p style="font-weight:bold; color:#333; margin-bottom:5px; text-align:right;">${item.category}: ${item.message}</p>

${item.suggestion ? `<pre style="background-color:#f8f8f8; padding:8px; border-radius:4px; font-size:0.85em; overflow-x:auto; text-align:left;"><code>${item.suggestion}</code></pre>` : ''}

</div>

`;

});

} else {

htmlOutput += `<p style="color:#666; text-align:center;">لا توجد تغذية راجعة تفصيلية حاليًا لهذا النوع من المراجعة.</p>`;

}

GA\_Helpers.logOperation('GAssistant.AI.ScriptDoctor', `getReviewFeedbackForSidebar:success`, { reviewType, feedbackCount: reviewResult.feedback.length });

return { html: htmlOutput, rawData: reviewResult }; // إرجاع HTML والبيانات الخام

} catch (e) {

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

return { html: `<p style="color:red; text-align:center;">🚨 حدث خطأ أثناء جلب المراجعة: ${e.message}</p>`, rawData: null };

}

}

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

// دوال لتوليد اختبارات الوحدات وفحص الأمان (للتوسعات المستقبلية)

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

/\*\*

\* @function generateUnitTestSuggestions

\* @description

\* 🧪 يطلب من نموذج Gemini توليد اقتراحات لاختبارات الوحدات لكود معين.

\* (هذه دالة هيكلية تحتاج إلى تكامل أعمق مع وحدة unit\_tests\_generator.gs لاحقًا).

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

\* @returns {object} - كائن يحتوي على أفكار الاختبارات.

\*/

function generateUnitTestSuggestions(code) {

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

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

// أو التفاعل مباشرة مع وحدة unit\_tests\_generator.gs عندما تكون جاهزة.

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

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

return result;

}

/\*\*

\* @function performSecurityScan

\* @description

\* 🔒 يطلب من نموذج Gemini إجراء فحص أمني لكود معين.

\* (هذه دالة هيكلية تحتاج إلى تكامل أعمق مع وحدة tools\_security\_review.gs لاحقًا).

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

\* @returns {object} - كائن يحتوي على المخاوف الأمنية.

\*/

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,

getReviewFeedbackForSidebar,

generateUnitTestSuggestions,

performSecurityScan

};

})(

GAssistant.Helpers,

GAssistant.AI.Dialogue,

GAssistant.AI.Core,

GAssistant.SettingsProvider // تأكد من تمرير GAssistant.SettingsProvider 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);

}