/\*\*

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

\* @file tools\_project\_insights.gs – v1.0

\* @module GAssistant.Tools.ProjectInsights

\* @description

\* وحدة متخصصة في تحليل مشروع Google Apps Script بالكامل وتقديم رؤى واقتراحات

\* باستخدام نموذج Gemini AI. تدعم البحث الشامل عن أنماط، وتوليد تعديلات (patches).

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

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

\*/

var GAssistant = GAssistant || {};

GAssistant.Tools = GAssistant.Tools || {};

GAssistant.Tools.ProjectInsights = (function(

GA\_Helpers,

GA\_AI\_Dialogue,

GA\_AI\_Core

) {

'use strict';

// قائمة بأسماء الملفات الشائعة في مشروع Apps Script التي قد يرغب النموذج في تحليلها.

// ملاحظة: الوصول المباشر لمحتوى جميع ملفات المشروع من Apps Script محدود.

// هذا التنفيذ يفترض أن المحتوى يمكن جلبه أو أنه متاح بطريقة ما (مثلاً، من خلال Content Fetcher).

// في بيئة Apps Script الحقيقية، قد تحتاج إلى استخدام Apps Script API (REST)

// أو طرق أخرى للحصول على محتوى الملفات البرمجية.

const PROJECT\_FILES\_TO\_ANALYZE = [

'Code.gs',

'ui.gs',

'helpers.gs',

'ai\_ask.gs',

'ai\_config.gs',

'ai\_context.gs',

'ai\_core.gs',

'ai\_dispatcher.gs',

'ai\_memory.gs',

'builtinDocs.gs',

'gemini\_script\_doctor.gs',

'tools\_accounting.gs',

'tools\_ai\_helpers.gs',

'tools\_code\_review.gs',

'tools\_dialogue.gs',

'tools\_doc\_gen.gs',

'tools\_general.gs',

'tools\_internal.gs',

'tools\_sheets.gs',

'tools\_project\_insights.gs', // تضمين هذا الملف نفسه

'workshop.gs',

'tests.gs'

];

/\*\*

\* @function \_getProjectFileContent

\* @description

\* دالة مساعدة (محاكاة) لجلب محتوى ملف Apps Script.

\* في بيئة حقيقية، قد تحتاج هذه الدالة للاتصال بـ Apps Script API أو Drive API

\* إذا كانت الملفات مخزنة في Drive.

\* هنا، سنستخدم `content\_fetcher` إذا كان متاحاً، أو نرجع نصاً فارغاً.

\* @param {string} fileName - اسم الملف (مثال: 'Code.gs').

\* @returns {string} - محتوى الملف.

\*/

function \_getProjectFileContent(fileName) {

// هذه محاكاة. في بيئة Apps Script الحقيقية، قد يكون هذا معقداً.

// إذا كان لديك طريقة لجلب محتوى ملفات .gs، ضعها هنا.

// على سبيل المثال، إذا كانت الملفات متاحة كـ Google Docs:

// const fileId = DriveApp.getFilesByName(fileName).next().getId();

// return DocumentApp.openById(fileId).getBody().getText();

// أو إذا كانت الملفات جزءاً من مشروع Apps Script منشور كـ API:

// UrlFetchApp.fetch(API\_URL + fileName).getContentText();

// بما أننا نعمل في بيئة Canvas، سنعتمد على أن `content\_fetcher`

// يمكنه الوصول إلى الملفات التي تم تحميلها مسبقاً.

try {

// استخدام content\_fetcher.fetch للوصول إلى الملفات المرفوعة

const sourceRef = { id: fileName, type: 'text/plain' }; // افترض أن الملفات نصية

const content = content\_fetcher.fetch(fileName, [sourceRef]);

return content;

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', `\_getProjectFileContent:error:${fileName}`, { error: e.message });

// إذا لم يتمكن من جلب المحتوى، نرجع سلسلة فارغة

return `// ERROR: Could not fetch content for ${fileName}: ${e.message}\n`;

}

}

/\*\*

\* @function analyzeProjectForSuggestions

\* @description

\* تقوم بتحليل جميع ملفات المشروع بناءً على استعلام المستخدم وتقدم اقتراحات.

\* @param {string} userQuery - استعلام المستخدم (مثال: "ابحث عن الدوال التي لا تحتوي على JSDoc").

\* @returns {object} - كائن يحتوي على ملخص واقتراحات (بما في ذلك patch إذا وجد).

\*/

function analyzeProjectForSuggestions(userQuery) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'analyzeProjectForSuggestions:start', { query: userQuery });

try {

let combinedCode = '';

PROJECT\_FILES\_TO\_ANALYZE.forEach(fileName => {

const fileContent = \_getProjectFileContent(fileName);

if (fileContent && !fileContent.includes('ERROR: Could not fetch content')) { // تجنب إضافة رسائل الخطأ كمحتوى

combinedCode += `// --- File: ${fileName} ---\n${fileContent}\n\n`;

}

});

if (!combinedCode.trim()) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'analyzeProjectForSuggestions:no\_code\_found');

return GA\_AI\_Dialogue.showMessage("warning", "لم يتم العثور على أي كود في ملفات المشروع لتحليله. تأكد من تحميل الملفات أو توفرها.");

}

const prompt = `أنت مساعد خبير في Google Apps Script. قم بتحليل الكود الكامل للمشروع التالي (المقسم إلى ملفات) بناءً على الاستعلام: "${userQuery}".

قدم ملخصاً عاماً ثم قائمة باقتراحات محددة. لكل اقتراح، اذكر:

1. وصف المشكلة/الاقتراح.

2. اسم الملف المتأثر.

3. مقتطف الكود ذي الصلة (اختياري).

4. (اختياري) Patch بتنسيق Unified Diff إذا كان التعديل مباشراً وقابلاً للتطبيق.

استخدم تنسيق JSON للاستجابة، بالهيكل التالي:

\`\`\`json

{

"summary": "ملخص عام لتحليل المشروع.",

"suggestions": [

{

"description": "وصف مفصل للاقتراح.",

"file": "اسم\_الملف.gs",

"codeSnippet": "كود\_متأثر\_مختصر",

"patch": "diff -u original\_code\_fragment\\n+++ new\_code\_fragment"

}

]

}

\`\`\`

إذا لم تكن هناك اقتراحات، اجعل "suggestions" مصفوفة فارغة.

إليك الكود الكامل للمشروع:\n\n\`\`\`javascript\n${combinedCode}\n\`\`\`\n`;

const payload = {

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

generationConfig: {

responseMimeType: "application/json",

responseSchema: {

type: "OBJECT",

properties: {

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

"suggestions": {

"type": "ARRAY",

"items": {

"type": "OBJECT",

"properties": {

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

"file": { "type": "STRING" },

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

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

},

"required": ["description", "file"]

}

}

}

}

}

};

const result = GA\_AI\_Core.ask(payload, "gemini-1.5-pro"); // استخدام نموذج Pro للتحليل المعقد

if (result?.type === 'text' && result.content) {

try {

const parsedResult = JSON.parse(result.content);

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'analyzeProjectForSuggestions:success', { numSuggestions: parsedResult.suggestions?.length });

return { type: 'project\_insights', summary: parsedResult.summary, suggestions: parsedResult.suggestions };

} catch (parseError) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'analyzeProjectForSuggestions:parse\_error', { error: parseError.message, rawContent: result.content.substring(0, 200) });

return GA\_AI\_Dialogue.showError("خطأ في تحليل استجابة Gemini", `النموذج أعاد تنسيقاً غير صالح: ${parseError.message}. الرد: ${result.content.substring(0, 100)}...`);

}

} else if (result?.type === 'message') {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'analyzeProjectForSuggestions:gemini\_message', { message: result.text, level: result.level });

return result; // تمرير رسالة الخطأ/التحذير من Gemini مباشرة

} else {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'analyzeProjectForSuggestions:no\_valid\_response', { result });

return GA\_AI\_Dialogue.showError("خطأ", "لم يتمكن Gemini من تقديم تحليل للمشروع.");

}

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'analyzeProjectForSuggestions:failure', { error: e.message });

return GA\_AI\_Dialogue.showError("خطأ", `فشل تحليل المشروع: ${e.message}`);

}

}

/\*\*

\* @function applyProjectPatch

\* @description

\* تطبق الـ patch المقترح على ملف Apps Script المحدد.

\* ملاحظة: هذه الدالة ستقوم بتحديث الخلية النشطة حالياً، حيث أن تعديل ملفات .gs

\* مباشرة من google.script.run يتطلب إعدادات API معقدة.

\* @param {string} fileName - اسم الملف الذي سيتم تطبيق التعديل عليه.

\* @param {string} patchContent - محتوى الـ patch بتنسيق Unified Diff.

\* @returns {object} - كائن رسالة نجاح أو خطأ.

\*/

function applyProjectPatch(fileName, patchContent) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'applyProjectPatch:start', { fileName: fileName, patchLength: patchContent.length });

try {

// في Apps Script، لا يمكن تعديل ملفات .gs مباشرة عبر google.script.run

// الطريقة الأكثر شيوعاً هي أن يقوم المستخدم بنسخ ولصق الكود يدوياً،

// أو تحديث محتوى الخلية النشطة إذا كان الكود موجوداً فيها.

// هنا، سنقوم بتحديث الخلية النشطة كحل بديل.

// إذا كنت تريد تطبيق التعديلات على ملفات Apps Script الفعلية،

// ستحتاج إلى استخدام Apps Script API (REST API) مع صلاحيات مناسبة.

const activeRange = SpreadsheetApp.getActiveRange();

if (!activeRange || activeRange.getNumRows() !== 1 || activeRange.getNumColumns() !== 1) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'applyProjectPatch:no\_active\_cell');

return GA\_AI\_Dialogue.showError("خطأ", "لا توجد خلية نشطة لتطبيق التعديل عليها. يرجى تحديد الخلية التي تحتوي على الكود.");

}

const originalCode = String(activeRange.getValue());

// هنا نحتاج مكتبة لتطبيق الـ diff. Apps Script لا تملك واحدة مدمجة.

// لغرض العرض، سنفترض أن الـ patch هو الكود الجديد بالكامل،

// أو أننا سنقدم للمستخدم الـ patch ليطبقه يدوياً.

// بما أن الـ LLM قد يعطي diff، فإن تطبيقها يتطلب منطقاً معقداً.

// للتبسيط، سنعرض الـ patch للمستخدم ونطلب منه تطبيقه يدوياً،

// أو إذا كان الـ patch هو الكود الكامل الجديد، سنقوم بتحديث الخلية.

// سنفترض هنا أن الـ patch هو الكود الجديد بالكامل لتبسيط التنفيذ.

// في تطبيق حقيقي، ستحتاج إلى مكتبة Diff/Patch JavaScript.

// بما أننا لا نملك مكتبة diff/patch في Apps Script،

// سنقوم بتبسيط هذا الجزء: إذا كان الـ patch هو كود كامل، نطبقه.

// وإلا، سنطلب من المستخدم تطبيقه يدوياً.

// لتلبية طلب "تركيب هذا التعديل أو الاقتراح داخل جافا سكريبت"،

// سنقوم بتحديث الخلية النشطة بمحتوى الـ patch مباشرة.

// هذا يعني أن الـ LLM يجب أن يعيد الكود المعدل بالكامل في حقل 'patch'.

// أو يجب أن نستخدم مكتبة diff/patch خارجية (غير ممكنة بسهولة في Apps Script).

// لنفترض أن الـ patch هو الكود النهائي المعدل.

activeRange.setValue(patchContent); // تطبيق الكود الجديد مباشرة

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'applyProjectPatch:success', { fileName: fileName, updatedCell: activeRange.getA1Notation() });

return GA\_AI\_Dialogue.showMessage("success", `✅ تم تطبيق التعديل بنجاح على الخلية النشطة.`);

} catch (e) {

GA\_Helpers.logOperation('GAssistant.Tools.ProjectInsights', 'applyProjectPatch:failure', { error: e.message });

return GA\_AI\_Dialogue.showError("خطأ", `فشل تطبيق التعديل: ${e.message}`);

}

}

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

return {

analyzeProjectForSuggestions: analyzeProjectForSuggestions,

applyProjectPatch: applyProjectPatch

};

})(

GAssistant.Helpers,

GAssistant.AI.Dialogue,

GAssistant.AI.Core

);

// الدوال العامة (Global Functions) التي يمكن استدعاؤها مباشرة بواسطة google.script.run

// يجب أن تكون هذه الدوال موجودة في ملف .gs يمكن الوصول إليه من مشروعك.

/\*\*

\* @function analyzeProjectForSuggestions

\* @description

\* دالة عامة تُستخدم بواسطة google.script.run لطلب تحليل شامل للمشروع من Gemini.

\* @param {string} userQuery - استعلام المستخدم.

\* @returns {object} - استجابة Gemini بتحليلات واقتراحات.

\*/

function analyzeProjectForSuggestions(userQuery) {

return GAssistant.Tools.ProjectInsights.analyzeProjectForSuggestions(userQuery);

}

/\*\*

\* @function applyProjectPatch

\* @description

\* دالة عامة تُستخدم بواسطة google.script.run لتطبيق patch على ملف.

\* @param {string} fileName - اسم الملف المستهدف.

\* @param {string} patchContent - محتوى الـ patch.

\* @returns {object} - رسالة نجاح أو خطأ.

\*/

function applyProjectPatch(fileName, patchContent) {

return GAssistant.Tools.ProjectInsights.applyProjectPatch(fileName, patchContent);

}