/\*\*

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

\* @file prompt\_engine.gs – v1.1

\* @module PROMPT\_ENGINE

\* @description

\* وحدة مساعدة لإنشاء systemPrompt ذكي لنموذج Gemini، تتضمن:

\* • وصف الورقة النشطة بصيغ Markdown أو JSON

\* • تخصيص نوع المهمة، الأسلوب، وتعليمات الاستخدام

\* • معالجة مسبقة لمدخلات المستخدم

\* • تكامل مع وحدات ضغط السياق

\* • التحقق من طول الـ Prompt

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

\*/

var PROMPT\_ENGINE = (function (GA\_Tools\_Sheets, GA\_ContextPacker, GA\_Helpers, GA\_AI\_Dialogue) { // ✅ تم تحديث التبعيات

'use strict';

/\*\*

\* @function \_cleanPromptText

\* @description

\* تقوم بتنظيف نص الـ Prompt من الرموز التعبيرية الزائدة والمسافات البيضاء.

\* يمكن توسيعها لاحقًا لإزالة العبارات المتكررة أو تطبيع النص.

\* @param {string} text - النص المراد تنظيفه.

\* @returns {string} - النص النظيف.

\*/

function \_cleanPromptText(text) {

// إزالة الرموز التعبيرية (قد تحتاج لقائمة أوسع من الـ unicode ranges)

let cleanedText = text.replace(/[\u{1F600}-\u{1F64F}\u{1F300}-\u{1F5FF}\u{1F680}-\u{1F6FF}\u{1F1E0}-\u{1F1FF}]/gu, '');

// إزالة المسافات البيضاء الزائدة والسطور الفارغة

cleanedText = cleanedText.replace(/\s+/g, ' ').trim();

GA\_Helpers.logOperation('PROMPT\_ENGINE', '\_cleanPromptText', { originalLength: text.length, newLength: cleanedText.length });

return cleanedText;

}

/\*\*

\* @function \_inferStyleFromPrompt

\* @description

\* 🧠 دالة وهمية: تستنتج الأسلوب من نص الـ prompt إذا لم يتم تحديده.

\* في تطبيق حقيقي، قد تستدعي نموذج AI أو تستخدم قواعد heuristic.

\* @param {string} userPrompt - نص طلب المستخدم.

\* @returns {string} - الأسلوب المستنتج أو سلسلة فارغة إذا لم يتم الاستنتاج.

\*/

function \_inferStyleFromPrompt(userPrompt) {

// هذا مجرد مثال بسيط. في تطبيق حقيقي، قد تستدعي نموذج AI هنا.

GA\_Helpers.logOperation('PROMPT\_ENGINE', '\_inferStyleFromPrompt:start', { promptPreview: userPrompt.substring(0, 50) });

let inferredStyle = '';

if (userPrompt.includes('تحليل')) {

inferredStyle = 'تحليلي';

} else if (userPrompt.includes('شرح') || userPrompt.includes('فهم')) {

inferredStyle = 'تعليمي';

} else if (userPrompt.includes('مساعدة') || userPrompt.includes('حل مشكلة')) {

inferredStyle = 'مساعد';

}

GA\_Helpers.logOperation('PROMPT\_ENGINE', '\_inferStyleFromPrompt:end', { inferredStyle });

return inferredStyle;

}

/\*\*

\* 🔧 يُنشئ systemPrompt مهيأ استنادًا إلى المعطيات

\* @param {Object} options

\* @param {string} options.taskType - نوع المهمة ("تحليل مالي"، "تلخيص"، إلخ)

\* @param {string} options.userPrompt - سؤال أو طلب المستخدم

\* @param {boolean} [options.includeSheet=true] - هل يتم تضمين وصف الورقة؟

\* @param {"markdown"|"json"|"none"} [options.sheetFormat="markdown"]

\* @param {string} [options.style] - أسلوب الكتابة المتوقع

\* @param {object} [options.modelConfig] - معلومات عن النموذج المستخدم (مثل { name: 'gemini-1.5-pro', version: '1.5' })

\* @returns {string} systemPrompt مهيأ للإرسال

\*/

function buildPrompt({

taskType = 'مهمة غير محددة',

userPrompt = '',

includeSheet = true,

sheetFormat = "markdown",

style = '',

modelConfig = null // ✅ معامل جديد

} = {}) {

GA\_Helpers.logOperation('PROMPT\_ENGINE', 'buildPrompt:start', { taskType, userPrompt: userPrompt.substring(0, 50) });

// ✅ 1. المعالجة المسبقة لـ userPrompt

userPrompt = \_cleanPromptText(userPrompt);

let prompt = `📌 المهمة: ${taskType}\n`;

// ✅ 2. توصيف الأسلوب تلقائيًا

if (!style) {

style = \_inferStyleFromPrompt(userPrompt); // استخدام الدالة المساعدة الداخلية

}

if (style) {

prompt += `🧭 الأسلوب المطلوب: ${style}\n`;

}

// ✅ 4. إضافة معرف للنموذج

if (modelConfig && modelConfig.name) {

prompt += `🤖 النموذج المستخدم: ${modelConfig.name}`;

if (modelConfig.version) {

prompt += ` (الإصدار: ${modelConfig.version})`;

}

prompt += `\n`;

}

if (includeSheet) {

const sheetInsight = getSheetDescription(sheetFormat);

if (sheetInsight) {

prompt += `\n${sheetInsight}\n`;

}

}

prompt += `\n🗣️ السؤال:\n${userPrompt}\n`;

// ✅ 5. عداد الطول

try {

GA\_Helpers.validatePromptLength(prompt); // التحقق من طول الـ Prompt النهائي

} catch (e) {

GA\_AI\_Dialogue.showError('خطأ في طول Prompt', e.message);

GA\_Helpers.logOperation('PROMPT\_ENGINE', 'buildPrompt:length\_validation\_failed', { error: e.message });

// يمكن هنا اتخاذ قرار بتقليص الـ prompt أو رمي خطأ وإيقاف العملية

// لغرض هذا المثال، سنسمح بالاستمرار بعد عرض الخطأ.

}

GA\_Helpers.logOperation('PROMPT\_ENGINE', 'buildPrompt:end', { finalPromptLength: prompt.length });

return prompt;

}

/\*\*

\* 🧠 استدعاء وصف الورقة بصيغة مناسبة

\* @param {"markdown"|"json"|"none"} format

\* @returns {string}

\*/

function getSheetDescription(format) {

GA\_Helpers.logOperation('PROMPT\_ENGINE', 'getSheetDescription:start', { format });

try {

let sheetInsight = "";

if (format === "markdown") {

sheetInsight = GA\_Tools\_Sheets.generateSheetAnalysisPrompt();

} else if (format === "json") {

sheetInsight = GA\_Tools\_Sheets.getSheetInsightAsJson();

}

// ✅ 3. التكامل مع packSheetInsight()

if (sheetInsight && typeof GA\_ContextPacker !== 'undefined' && GA\_ContextPacker.packSheetInsight) {

sheetInsight = GA\_ContextPacker.packSheetInsight(sheetInsight);

GA\_Helpers.logOperation('PROMPT\_ENGINE', 'getSheetDescription:packed', { originalLength: sheetInsight.length, newLength: sheetInsight.length }); // ContextPacker.packSheetInsight قد لا تغير الطول فعليًا

}

GA\_Helpers.logOperation('PROMPT\_ENGINE', 'getSheetDescription:end', { insightLength: sheetInsight.length });

return sheetInsight;

} catch (e) {

GA\_Helpers.logOperation('PROMPT\_ENGINE', 'getSheetDescription:error', { error: e.message });

GA\_AI\_Dialogue.showError('خطأ في وصف الورقة', `❌ فشل في جلب وصف الورقة: ${e.message}`);

return `❌ فشل في جلب وصف الورقة: ${e.message}`;

}

}

return {

buildPrompt

};

})(

GAssistant.Tools.Sheets, // ✅ حقن GAssistant.Tools.Sheets

GAssistant.ContextPacker, // ✅ حقن GAssistant.ContextPacker

GAssistant.Helpers, // ✅ حقن GAssistant.Helpers

GAssistant.AI.Dialogue // ✅ حقن GAssistant.AI.Dialogue

);