/\*\*

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

\* @file ai\_core.gs – v1.2

\* @module GAssistant.AI.Core

\* @description

\* يحتوي على كل الإعدادات والثوابت الخاصة بسلوك محرك الذكاء الاصطناعي.

\* يعتمد على الثوابت الأساسية من GAssistant.Config ويوفر إعدادات متقدمة للـ API.

\* يعتمد على: GAssistant.Config, GAssistant.Helpers, GAssistant.AI.Dispatcher, GAssistant.AI.Memory, GAssistant.AI.Dialogue, GAssistant.ContextPacker

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

\*/

var GAssistant = GAssistant || {};

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

// حقن التبعيات: يجب تمرير الوحدات المعتمدة هنا

GAssistant.AI.Core = (function(GA\_Config, GA\_Helpers, GA\_AI\_Dispatcher, GA\_AI\_Memory, GA\_AI\_Dialogue, GA\_ContextPacker) { // ✅ تم إضافة GA\_ContextPacker

'use strict';

/\*\*

\* @function ask

\* @description

\* الدالة الرئيسية للتفاعل مع نموذج Gemini.

\* تقوم ببناء الطلب، استدعاء API، ومعالجة الرد.

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

\* @param {string} [modelOverride] - اسم النموذج المراد استخدامه، يتجاوز الاختيار التلقائي.

\* @param {string} [imageBlob] - بيانات الصورة (Blob) إذا كان الطلب يتضمن صورة.

\* @returns {object} - كائن الرد النهائي من النموذج أو الأداة المنفذة.

\*/

function ask(userPrompt, modelOverride, imageBlob) {

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:start', { prompt: userPrompt.substring(0, 50), modelOverride, hasImage: !!imageBlob });

let state = {

userPrompt: userPrompt,

model: null,

systemInstruction: "",

history: [],

tools: [],

images: [],

final: null, // النتيجة النهائية التي ستُعرض للمستخدم

halt: false // إذا كان true، تتوقف دورة المعالجة

};

try {

// 1. تحديد النموذج المناسب

state.model = GA\_Config.getPreferredGeminiModel(userPrompt, modelOverride);

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:model\_selected', { model: state.model });

// 2. إثراء السياق (System Instruction)

// هذا الجزء يتم التعامل معه الآن بواسطة AI\_CONTEXT (يفترض أن AI\_CONTEXT.enrichContext تقوم بتحديث state.systemInstruction)

// ملاحظة: AI\_CONTEXT ليست تبعية مباشرة هنا، بل يتم استدعاء وظائفها من مكان آخر أو يجب أن تكون محقونة.

// لغرض هذا المثال، سنفترض أن AI\_CONTEXT.enrichContext متاحة في النطاق العام أو تم حقنها بشكل غير مباشر.

// إذا لم تكن AI\_CONTEXT محقونة، يجب إضافتها إلى تبعيات GAssistant.AI.Core

// For now, assuming GA\_AI\_Context is globally available or implicitly managed.

// If AI\_CONTEXT is a module, it should be passed as a dependency here.

// As per the provided ai\_ask.gs, GA\_AI\_Context is used without explicit import.

// Let's assume it's correctly handled in the broader Apps Script project setup.

if (typeof GA\_AI\_Context !== 'undefined' && GA\_AI\_Context.enrichContext) {

GA\_AI\_Context.enrichContext(state);

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:context\_enriched', { systemInstructionLength: state.systemInstruction.length });

} else {

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:context\_enrichment\_skipped', { reason: 'GA\_AI\_Context not available' });

// Fallback or error handling if AI\_CONTEXT is not available

// For a robust system, GA\_AI\_Context should be a dependency.

}

// 3. إضافة سجل المحادثة السابق

state.history = GA\_AI\_Memory.getSessionHistory();

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:history\_loaded', { historyLength: state.history.length });

// 4. إضافة الصورة إذا وجدت

if (imageBlob) {

state.images.push(imageBlob);

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:image\_added', {});

}

// 5. تهيئة الأدوات

state.tools = GA\_Tools\_Catalog.getFunctionList(); // افتراض أن GA\_Tools\_Catalog متاحة

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:tools\_loaded', { toolCount: state.tools.length });

// ✅ 6. تبسيط وتكثيف السياق باستخدام ContextPacker

state = GA\_ContextPacker.packContext(state);

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:context\_packed', { packedPromptLength: state.userPrompt.length });

// 7. استدعاء AI\_ASK لإرسال الطلب ومعالجة الردود

state = GA\_AI\_Ask.callGeminiAgent(state); // افتراض أن GA\_AI\_Ask متاحة

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:gemini\_agent\_called', { halt: state.halt, finalType: state.final ? state.final.type : 'none' });

// 8. حفظ آخر طلب للذاكرة (لإعادة التنفيذ)

GA\_AI\_Memory.setLastPrompt(userPrompt, state.model);

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:end', { finalType: state.final ? state.final.type : 'none' });

return state.final;

} catch (e) {

GA\_Helpers.logOperation('GAssistant.AI.Core', 'ask:exception', { error: e.message, stack: e.stack });

return GA\_AI\_Dialogue.showError(`💥 حدث خطأ غير متوقع في محرك الذكاء الاصطناعي: ${e.message}`);

}

}

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

return {

ask

};

})(

GAssistant.Config,

GAssistant.Helpers,

GAssistant.AI.Dispatcher, // افتراض وجود هذه الوحدة

GAssistant.AI.Memory, // افتراض وجود هذه الوحدة

GAssistant.AI.Dialogue, // افتراض وجود هذه الوحدة

GAssistant.ContextPacker // ✅ تم حقن GAssistant.ContextPacker

);