// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- START OF FILE: 10\_ui/1\_ui.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*

\* @file 10\_ui/1\_ui.gs

\* @module GAssistant.UI

\* @description

\* وحدة مسؤولة عن إنشاء وإدارة عناصر واجهة المستخدم.

\* تم تحويلها إلى دالة (Module Factory) تُستدعى بواسطة Initializer.

\*/

var GAssistant = GAssistant || {}; // تأكيد وجود الكائن الرئيسي GAssistant

GAssistant.UI = (() => { // هذا IIFE سيتم تنفيذه فورًا عند تحميل الملف 10\_ui/1\_ui.gs.

// يجب أن يتم تعيين ناتجه إلى GAssistant.UI في \_initializeGAssistantSystemDynamically().

'use strict';

// دالة مساعدة لـ Injection. سيتم استدعاؤها في كل دالة مصدرة (exported function).

function \_getInjectedDependencies() {

// Injector سيكون متاحًا لأن 02\_utils.gs يتم تحميله أبجديًا قبل هذا الملف.

if (!GAssistant?.Utils?.Injector) {

console.error("CRITICAL: GAssistant.Utils.Injector not ready in 10\_ui/1\_ui.gs.");

// رمي خطأ لوحدة Utils لتسجيله بشكل صحيح.

if (GAssistant.Utils && typeof GAssistant.Utils.error === 'function') {

GAssistant.Utils.error("UI: Injector not ready at \_getInjectedDependencies call.");

}

throw new Error("UI Module: Core dependencies not initialized.");

}

// جلب جميع التبعيات اللازمة لهذه الوحدة مرة واحدة.

return GAssistant.Utils.Injector.get('Utils', 'UI.Dialogue', 'Config', 'API', 'Tools', 'AI', 'UI');

}

/\*\*

\* دالة Apps Script القياسية التي تُنفذ عند فتح المستند.

\* تُنشئ القوائم المخصصة في Google Sheets.

\* @public

\*/

function onOpen() {

// \*\*التصحيح\*\*: جلب التبعيات داخل الدالة.

const { Utils } = \_getInjectedDependencies(); // Utils سيكون متاحاً هنا

try {

if (typeof SpreadsheetApp.getUi !== 'function') {

Utils.warn("UI.onOpen: SpreadsheetApp.getUi() غير متاح في هذا السياق.");

return;

}

const ui = SpreadsheetApp.getUi();

ui.createMenu('✨ G-Assistant')

.addItem('💬 فتح المساعد', 'showAssistantSidebar')

.addItem('👨‍💻 فتح أدوات المطور', 'showDeveloperSidebar')

.addItem('🌐 فتح لوحة التحكم (WebApp)', 'showWebApp')

.addSeparator()

.addItem('▶️ إرسال محتوى الخلية كأمر', 'sendPromptFromSelection')

.addSeparator()

.addSubMenu(ui.createMenu('⚙️ النظام')

.addItem('تصدير توثيق الأدوات', 'exportToolsDocumentationToDoc')

.addItem('تصدير آخر استجابة', 'exportLastResponseToDoc')

.addItem('إعداد/تحديث مؤقتات الوكلاء', 'setupAgentTriggersGlobally')

.addItem('مسح ذاكرة الجلسة', 'clearSessionContext')

.addItem('🧪 تشغيل الاختبارات', 'runAllTests')

)

.addToUi();

Utils.log('UI.onOpen: Custom menu created successfully.');

} catch (e) {

// \*\*التصحيح\*\*: استخدام Utils.error لتوحيد التسجيل.

Utils.error("UI.onOpen error: Failed to create menu.", e);

if (typeof SpreadsheetApp.getUi === 'function') {

SpreadsheetApp.getUi().alert("⚠️ فشل أثناء إنشاء القائمة. تحقق من تحميل الوحدات.");

}

}

}

/\*\*

\* دالة مساعدة داخلية لإنشاء وعرض شريط جانبي.

\* @private

\*/

function \_showSidebar(fileName, title, width) {

// \*\*التصحيح\*\*: جلب التبعيات داخل الدالة.

const { Utils } = \_getInjectedDependencies();

try {

const html = HtmlService.createHtmlOutputFromFile(fileName).setTitle(title).setWidth(width);

SpreadsheetApp.getUi().showSidebar(html);

Utils.log(`UI.\_showSidebar: Sidebar '${fileName}' opened.`);

} catch (e) {

Utils.error(`UI.\_showSidebar error: Failed to open sidebar '${fileName}'.`, e);

if (typeof SpreadsheetApp.getUi === 'function') {

SpreadsheetApp.getUi().alert(`فشل فتح الشريط الجانبي: ${e.message}`);

}

}

}

/\*\*

\* تعرض الشريط الجانبي الرئيسي للمساعد.

\* @public

\*/

function showAssistantSidebar() {

const { Config } = \_getInjectedDependencies();

\_showSidebar(Config.get('ASSISTANT\_SIDEBAR\_FILE'), '✨ مساعد G-Assistant', 350);

}

/\*\*

\* تعرض الشريط الجانبي لأدوات المطور.

\* @public

\*/

function showDeveloperSidebar() {

const { Config } = \_getInjectedDependencies();

\_showSidebar(Config.get('DEVELOPER\_SIDEBAR\_FILE'), '👨‍💻 أدوات المطور', 500);

}

/\*\*

\* تعرض لوحة تحكم تطبيق الويب في نافذة منبثقة (Modal Dialog).

\* @public

\*/

function showWebApp() {

const { Config, Utils } = \_getInjectedDependencies();

Utils.executeSafely(() => {

const html = HtmlService.createHtmlOutputFromFile(Config.get('WEB\_APP\_FILE'))

.setWidth(900)

.setHeight(700);

SpreadsheetApp.getUi().showModalDialog(html, 'لوحة تحكم G-Assistant');

}, [], 'showWebApp');

}

/\*\*

\* يعرض لوحة API Schema (مثلاً في شريط جانبي للمطور).

\* @public

\*/

function renderApiSchemaPanel() {

const { Utils, Dialogue, API } = \_getInjectedDependencies();

return Utils.executeSafely(() => {

// افتراض وجود دالة getSchema في وحدة API

const schemaResponse = API.getSchema ? API.getSchema() : Dialogue.createError("API.getSchema غير متوفرة.");

return schemaResponse?.type === 'success' && schemaResponse.data

? Dialogue.createSuccess("تم جلب توثيق API بنجاح.", schemaResponse.data)

: Dialogue.createError("فشل في جلب توثيق واجهة API.");

}, [], 'renderApiSchemaPanel');

}

/\*\*

\* يعرض لوحة ورشة عمل المطورين (محتويات ورقة Staging).

\* @public

\*/

function renderWorkshopPanel() {

const { Utils, Dialogue, Config } = \_getInjectedDependencies();

return Utils.executeSafely(() => {

const sheetName = Config.get('DEVELOPER\_WORKSHOP\_SHEET'); // يجب أن يكون هذا الإعداد في Config

const sheet = Utils.getSheet(sheetName);

if (!sheet || sheet.getLastRow() < 2) {

return Dialogue.createInfo("ورشة عمل المطورين فارغة حاليًا أو غير موجودة.");

}

const data = sheet.getRange(2, 1, sheet.getLastRow() - 1, sheet.getLastColumn()).getValues();

const headers = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

return Dialogue.createTable(`محتويات ورشة العمل (${data.length} عنصر)`, headers, data);

}, [], 'renderWorkshopPanel');

}

/\*\*

\* يعرض لوحة الرؤى والتحليلات الخاصة بالمشروع (مراجعة الكود، البنية).

\* @public

\*/

function renderInsightsPanel() {

const { Utils, Dialogue, Tools } = \_getInjectedDependencies();

return Utils.executeSafely(() => {

const result = Tools.ProjectInsights.analyzeProject({ userQuery: 'تحليل عام للبنية والتحسينات المحتملة' });

return result?.type === 'success'

? result

: Dialogue.createError("فشل تحليل المشروع أو عدم توفر الكود.");

}, [], 'renderInsightsPanel');

}

/\*\*

\* يرسل محتوى الخلية المحددة حاليًا إلى المساعد كأمر.

\* @public

\*/

function sendPromptFromSelection() {

const { Utils, UI } = \_getInjectedDependencies(); // UI هنا هي GAssistant.UI

const prompt = UI.getCodeFromActiveCell();

if (prompt) {

showAssistantSidebar(); // عرض الشريط الجانبي للمساعد

// يمكن هنا إرسال الـ prompt إلى حقل الإدخال في الشريط الجانبي

// أو استدعاء askGemini مباشرة. لتبسيط التفاعل، نكتفي بالتنبيه.

SpreadsheetApp.getUi().alert(`سيتم الآن إرسال محتوى الخلية كأمر:\n"${prompt.substring(0, 50)}..."`);

} else {

SpreadsheetApp.getUi().alert("يرجى تحديد خلية تحتوي على أمر أولاً.");

}

}

/\*\*

\* يصدّر آخر استجابة من المساعد إلى مستند Google Docs.

\* @public

\*/

function exportLastResponseToDoc() {

const { Utils, AI } = \_getInjectedDependencies();

Utils.executeSafely(() => {

const history = AI.Memory.getSessionHistory();

const lastResponse = history.filter(m => m.role === 'model').pop();

if (!lastResponse || !lastResponse.parts || !lastResponse.parts[0] || !lastResponse.parts[0].text) {

SpreadsheetApp.getUi().alert("لا توجد استجابة سابقة لحفظها.");

return;

}

const doc = DocumentApp.create(`استجابة G-Assistant - ${new Date().toLocaleString()}`);

doc.getBody().appendParagraph(lastResponse.parts[0].text);

SpreadsheetApp.getUi().alert(`✅ تم تصدير آخر استجابة بنجاح.\n📄 الرابط: ${doc.getUrl()}`);

}, [], 'exportLastResponseToDoc');

}

/\*\*

\* تجلب الكود من الخلية النشطة في Google Sheet.

\* @returns {string} - محتوى الخلية النشطة كنص.

\* @public

\*/

function getCodeFromActiveCell() {

const { Utils } = \_getInjectedDependencies();

try {

const activeRange = SpreadsheetApp.getActiveRange();

if (activeRange && activeRange.getNumRows() === 1 && activeRange.getNumColumns() === 1) {

return String(activeRange.getValue());

}

return '';

} catch (e) {

Utils.error('UI.getCodeFromActiveCell: Failed to get code from cell.', e);

return '';

}

}

/\*\*

\* تحدث الخلية النشطة بكود جديد (بعد تطبيق التعديل).

\* @param {string} newCode - الكود الجديد لتحديث الخلية به.

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

\* @public

\*/

function updateActiveCellWithCode(newCode) {

const { Utils, Dialogue } = \_getInjectedDependencies();

try {

const activeRange = SpreadsheetApp.getActiveRange();

if (activeRange && activeRange.getNumRows() === 1 && activeRange.getNumColumns() === 1) {

activeRange.setValue(newCode);

return Dialogue.createSuccess('✅ تم تحديث الخلية بالكود الجديد.');

}

return Dialogue.createError('❌ لا توجد خلية نشطة لتحديثها.');

} catch (e) {

Utils.error('UI.updateActiveCellWithCode: Failed to update cell.', e);

return Dialogue.createError(`🚨 فشل تحديث الخلية: ${e.message}`);

}

}

return {

onOpen,

showAssistantSidebar,

showDeveloperSidebar,

showWebApp,

renderApiSchemaPanel,

renderWorkshopPanel,

renderInsightsPanel,

sendPromptFromSelection,

exportLastResponseToDoc,

getCodeFromActiveCell, // يجب تصديرها ليتمكن Code.gs من الوصول إليها

updateActiveCellWithCode // يجب تصديرها ليتمكن Code.gs من الوصول إليها

};

})();

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// --- END OF FILE: 10\_ui/1\_ui.gs ---

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*