/\*\*

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

\* @file tools\_accounting.gs

\* @module GAssistant.Tools.Accounting

\* @description

\* أدوات المحاسبة والتحليل المالي.

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

\*/

var GAssistant = GAssistant || {};

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

GAssistant.Tools.Accounting = (() => {

'use strict';

// --- \*\*التصحيح\*\*: تم تغيير طريقة تعريف التبعيات ---

// بدلاً من تفكيكها في الأعلى، سنشير إليها كمتغيرات للوحدات الكاملة.

const Utils = GAssistant.Utils;

const Config = GAssistant.Config;

// سنستدعي Dialogue مباشرة عند الحاجة لضمان تحميله.

/\*\*

\* يحسب إجمالي الربح لفترة زمنية محددة.

\* @param {object} args - { startDate, endDate }.

\* @returns {object} - كائن يحتوي على جدول ملخص أو رسالة خطأ.

\*/

function calculateGrossProfit(args) {

const { startDate, endDate } = args;

return Utils.executeSafely(() => {

const config = Config.getAll();

const revenueSheet = Utils.getSheet(config.REVENUE\_SHEET);

const expensesSheet = Utils.getSheet(config.EXPENSES\_SHEET);

const totalRevenue = \_calculateTotal(revenueSheet, startDate, endDate, 0, 5);

const totalExpenses = \_calculateTotal(expensesSheet, startDate, endDate, 0, 2);

const grossProfit = totalRevenue - totalExpenses;

const currency = config.DEFAULT\_CURRENCY;

Utils.log('Tools.Accounting.calculateGrossProfit: Success', { totalRevenue, totalExpenses, grossProfit });

const headers = ["المقياس", "القيمة"];

const rows = [

["إجمالي الإيرادات", `${totalRevenue.toFixed(2)} ${currency}`],

["إجمالي المصروفات", `${totalExpenses.toFixed(2)} ${currency}`],

["صافي الربح الإجمالي", `${grossProfit.toFixed(2)} ${currency}`]

];

// \*\*التصحيح\*\*: استدعاء الدالة بالمسار الكامل

return GAssistant.UI.Dialogue.createTable(`ملخص الربح من ${startDate} إلى ${endDate}`, headers, rows);

}, [], `calculating gross profit for ${startDate} to ${endDate}`);

}

/\*\*

\* دالة مساعدة لحساب الإجمالي من ورقة معينة ضمن نطاق تاريخي.

\* @private

\*/

function \_calculateTotal(sheet, startDate, endDate, dateColIdx, amountColIdx) {

if (!sheet || sheet.getLastRow() < 2) return 0;

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

const start = new Date(startDate);

const end = new Date(endDate);

let total = 0;

data.forEach(row => {

try {

const rowDate = new Date(row[dateColIdx]);

if (rowDate >= start && rowDate <= end) {

total += Number(row[amountColIdx]) || 0;

}

} catch(e) { /\* تجاهل التواريخ غير الصالحة \*/ }

});

return total;

}

/\*\*

\* يولد ملخصًا ماليًا لليوم المحدد.

\* @param {object} args - { date: string }.

\* @returns {object} - كائن رسالة.

\*/

function generateDailySummary(args) {

const { date } = args;

const targetDate = date || Utilities.formatDate(new Date(), Session.getScriptTimeZone(), "yyyy-MM-dd");

return Utils.executeSafely(() => {

const resultTableResponse = calculateGrossProfit({ startDate: targetDate, endDate: targetDate });

if(resultTableResponse.type !== 'table') {

// \*\*التصحيح\*\*: استدعاء الدالة بالمسار الكامل

return GAssistant.UI.Dialogue.createError(`فشل حساب ملخص يوم ${targetDate}.`);

}

const summaryData = resultTableResponse.data.rows.reduce((acc, row) => {

acc[row[0]] = row[1];

return acc;

}, {});

const summarySheet = Utils.getSheet(Config.get('DAILY\_SUMMARY\_SHEET'), ["التاريخ", "الإيرادات", "المصروفات", "صافي الربح", "العملة"]);

const newRow = [

new Date(targetDate),

parseFloat(summaryData["إجمالي الإيرادات"]) || 0,

parseFloat(summaryData["إجمالي المصروفات"]) || 0,

parseFloat(summaryData["صافي الربح الإجمالي"]) || 0,

Config.get('DEFAULT\_CURRENCY')

];

summarySheet.appendRow(newRow);

Utils.log('Tools.Accounting.generateDailySummary: Success', { date: targetDate });

// \*\*التصحيح\*\*: استدعاء الدالة بالمسار الكامل

return GAssistant.UI.Dialogue.createTable(`تم إنشاء/تحديث الملخص اليومي لتاريخ ${targetDate}`, resultTableResponse.data.headers, resultTableResponse.data.rows);

}, [], `generating daily summary for ${targetDate}`);

}

/\*\*

\* يسجل إدخالاً جديداً (إيراد أو مصروف) في الورقة المناسبة.

\* @param {object} args - { type: 'revenue'|'expense', data: object }

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

\*/

function logEntry(args) {

const { type, data } = args;

return Utils.executeSafely(() => {

const config = Config.getAll();

const now = new Date();

if(type === 'revenue') {

const sheet = Utils.getSheet(config.REVENUE\_SHEET, ["التاريخ", "المنتج", "الكمية", "سعر الوحدة", "الخصم", "الإجمالي"]);

const total = (Number(data.quantity) || 0) \* (Number(data.unitPrice) || 0);

sheet.appendRow([now, data.product, data.quantity, data.unitPrice, 0, total]);

// \*\*التصحيح\*\*: استدعاء الدالة بالمسار الكامل

return GAssistant.UI.Dialogue.createSuccess(`✅ تم تسجيل إيراد للمنتج "${data.product}" بقيمة ${total.toFixed(2)}.`);

} else if (type === 'expense') {

const sheet = Utils.getSheet(config.EXPENSES\_SHEET, ["التاريخ", "الوصف", "المبلغ", "الفئة"]);

sheet.appendRow([now, data.description, data.amount, data.category]);

// \*\*التصحيح\*\*: استدعاء الدالة بالمسار الكامل

return GAssistant.UI.Dialogue.createSuccess(`✅ تم تسجيل مصروف "${data.description}" بقيمة ${Number(data.amount).toFixed(2)}.`);

} else {

// \*\*التصحيح\*\*: استدعاء الدالة بالمسار الكامل

return GAssistant.UI.Dialogue.createError(`نوع الإدخال "${type}" غير معروف. استخدم 'revenue' أو 'expense'.`);

}

}, [], `logging new ${type} entry`);

}

return {

calculateGrossProfit,

generateDailySummary,

logEntry,

};

})();