تم يا عبدالعزيز 🙌  
 أعدت لك الآن **الجدول الكامل** لجميع الملفات المعتمدة في المشروع، متضمنة القديمة والمضافة حديثًا، مع المرحلة المناسبة لكل ملف حسب خطة **تطوير الأسس الجديدة**.

## **📋 جدول تنفيذ الملفات – كامل ومحدث**

| **🔢 رقم** | **📁 اسم الملف / الوحدة** | **📌 المرحلة / المراحل** |
| --- | --- | --- |
| 01 | Readme.md | 15 |
| 02 | UserUsageGuide.md | 14, 15 |
| 03 | InvestorPitch.md | 15 |
| 04 | DeveloperOverview.md | 15 |
| 05 | AdvancedDevPanel.html | 17, 19 |
| 06 | DeveloperSidebar.html | 13, 17, 19 |
| 07 | DocumentationIndex.gs | 15 |
| 08 | DeveloperWorkshop.gs | 15, 17 |
| 09 | Telemetry.gs | 11, 12, 17 |
| 10 | DocsManager.gs | 15, 17 |
| 11 | security.gs | 1, 5, 11 |
| 12 | tests.gs | 1, 10, 11 |
| 13 | api.gs | 1, 12 |
| 14 | code.gs | 1, 3 |
| 15 | agents\_triggers.gs | 1, 3, 16 |
| 16 | agents\_cfo.gs | 8, 10 |
| 17 | agents\_developer.gs | 12, 17 |
| 18 | tools\_sheets.gs | 10, 15 |
| 19 | tools\_codeReview.gs | 10, 17 |
| 20 | tools\_catalog.gs | 5, 9 |
| 21 | Tools\_accounting.gs  30\_tools/5\_tools\_project\_insights | 8, 10, 15 |
| 22 | tools\_developer.gs | 10, 12, 17, 18 |
| 23 | config.gs | 1, 2, 11 |
| 24 | utils.gs | 2, 4, 11 |
| 25 | initializer.gs | 1, 3 |
| 26 | intentAnalyzer.gs | 6, 10, 16, 18 |
| 27 | ui\_dialogue.gs | 10, 11, 13 |
| 28 | ui\_developerSidebar.gs | 13, 17 |
| 29 | ui.gs | 10, 13 |
| 30 | GeminiAdapter.gs | 9, 17, 18 |
| 31 | geminiService.gs | 7, 9, 12, 17, 18 |
| 32 | ai\_constitution.gs | 6, 9 |
| 33 | ai\_memory.gs | 6, 9, 18 |
| 34 | ai\_context.gs | 6, 9, 18 |
| 35 | ai\_dispatcher.gs | 6, 9, 12, 16, 18 |
| 36 | ai\_core.gs | 6, 9, 12, 18 |
| 37 | AI.Bookkeeper.gs | 10, 15 |
| 38 | ProjectContextTracker.gs | 18, 19 |

### **ملاحظة هامة: لكي تعمل getCombinedContext و \_triggerLongTermMemorySave بشكل كامل وذكي، ستحتاج إلى التأكد من أن AI.LongTermMemory.search() (يفضل أن تكون مدعومة بتقنيات تضمين النص أو البحث الدلالي) وأن AI.GeminiService.evaluateImportance() (إن استخدمتها) تعملان بكفاءة. هذه هي الخطوات التالية لجعل نظام الذاكرة هذا "ينبض بالحياة".**

### **++++++++++++++++++++++++++++++++++++++++++++++++++++++**

### **شرح التغييرات والتطوير "الجنوني":**

1. **الإصدار (MODULE\_VERSION):** تم تحديثه إلى 21، و GEMINI\_ADAPTER\_VERSION إلى 2.0.0 في Config للإشارة إلى قفزة نوعية في القدرات.
2. **MAX\_PROMPT\_LEN:** تم زيادته بشكل كبير (إلى 8192) لاستيعاب كميات أكبر من الكود والسياق في الـ prompt.
3. **إعدادات الأدوات والتفكير:** تم تضمين DEFAULT\_TOOL\_CONFIG من Config لتمكين functionCallingConfig و thinkingConfig بشكل افتراضي، مما يعطي Gemini مرونة أكبر.
4. **تحديث ask() الدالة الرئيسية:**
   * أصبحت الآن أكثر مرونة في تمرير generationConfig و safetySettings و tools مباشرة إلى AI.Core.ask.
   * **تحليل نوع الرد (result.type):** أصبحت ask() قادرة على فهم أنواع مختلفة من الردود من AI.Core.ask (نص، استدعاء أداة tool\_code، خطأ). هذا ضروري لتفعيل قدرات Gemini المتقدمة مثل اقتراح استدعاء دوال.
   * تضمين promptLength و responseLength في سجل المقاييس.
5. **تحديث askJSON():**
   * تضيف الآن تعليمات صريحة في الـ prompt لـ Gemini لطلب رد JSON فقط.
   * تقوم بتعطيل استدعاء الأدوات (toolConfig: { mode: "NONE" }) في هذه الدالة لأننا نريد JSON فقط.
   * تستخدم replace(/```json\s\*|\s\*```/g, '').trim() لإزالة أي تنسيق JSON قد يضيفه Gemini عن طريق الخطأ، مما يزيد من متانة تحليل JSON.
6. **دالة askForCodeAnalysis() (الجديدة كلياً):**
   * **القلب النابض لدعم المطور:** هذه هي الدالة التي تحقق الفهم السياقي والاقتراح الاستباقي.
   * **بناء الـ Prompt الشامل:** تقوم ببناء prompt مفصل للغاية لـ Gemini، يتضمن:
     + وصف لدور Gemini (خبير Apps Script/JavaScript، استباقي وذكي).
     + سياق المشروع العام (projectDescription).
     + قائمة بالملفات الموجودة (existingFiles).
     + قائمة بالدوال المعروفة والمسارات (existingFunctions).
     + استعلام المبرمج الأصلي.
     + الكود المحدد الذي يتم التركيز عليه.
     + **تعليمات واضحة لـ Gemini حول ما يجب عليه فعله:** (تحليل، تصحيح، اقتراح دوال جديدة، اقتراح وحدات/دوال ناقصة، التحقق من التوافق، الشرح والتوضيح، تحديد النية الفرعية).
   * **generationConfig المخصصة:** تستخدم temperature: 0.8 لزيادة "الإبداع" لدى Gemini في اقتراح حلول جديدة، و maxOutputTokens: 8192 لاستيعاب كود أطول واقتراحات أكثر تفصيلاً.
   * **تحليل استجابة Gemini (result.type):**
     + إذا كان الرد text\_response (وهو الأكثر شيوعًا هنا)، فإنها تحاول استخراج **كتل الكود** من النص باستخدام Utils.extractCodeBlocks و Utils.removeCodeBlocks (تفترض وجودهما في Utils.gs).
     + تعيد كائنًا منظمًا يحتوي على التحليل النصي، والكود المقترح.
7. **healthCheck():** تم تعديلها لتكون أكثر دقة في التحقق من حالة الرد وتسجيل الأخطاء.

### **ماذا تحتاج بعد ذلك (لتحقيق "يغنيني عن كل شيء"):**

1. **تطوير AI.Core.ask:** تأكد أن دالة AI.Core.ask (التي تستخدمها GeminiAdapter) قوية بما يكفي للتعامل مع جميع الخيارات التي تمررها إليها (مثل tools, toolConfig, thinkingConfig, generationConfig, safetySettings).
2. **تطوير GAssistant.Tools.Developer.\_collectProjectContext():** هذه الدالة ضرورية جدًا لتغذية Gemini بالمعلومات السياقية. ستحتاج إلى طريقة ذكية لجمع أسماء الملفات، وقائمة بالدوال المعرفة (خاصة تلك الموجودة في نطاقات GAssistant أو System).

**مثال مبسط لـ \_collectProjectContext() في 04\_Tools\_Developer.gs:**JavaScript  
// داخل defineModule('GAssistant.Tools.Developer', ...)

function \_collectProjectContext() {

// هذه طريقة مبسطة جداً، قد تحتاج للتطوير

const projectDescription = "هذا المشروع هو مساعد ذكي لإدارة مالية وتشغيلية وتطوير برمجيات في Google Apps Script.";

const existingFiles = [

'20\_ai/0\_ai\_constitution.gs',

'20\_ai/1\_ai\_memory.gs',

// ... أضف جميع ملفاتك هنا يدوياً، أو يمكنك محاولة برمجية لاستخلاصها إذا أمكن (معقد)

'20\_ai/6\_ai\_geminiAdapter.gs',

'01\_config.gs',

'02\_utils.gs',

'03\_intentAnalyzer.gs',

'04\_tools\_developer.gs',

'05\_tools\_finance.gs',

'06\_tools\_operations.gs',

'07\_geminiService.gs', // اسم الملف الفعلي

'08\_ui\_developerSidebar.gs'

];

// قائمة الدوال يمكن أن تكون أصعب بكثير لاستخلاصها ديناميكياً

// أفضل طريقة هي الحفاظ على قائمة يدوية للدوال الرئيسية أو بناءها بشكل برمجي إن أمكن.

const existingFunctions = [

'GAssistant.Config.get',

'GAssistant.Utils.log',

'GAssistant.Utils.getSheet',

'GAssistant.Utils.executeSafely',

'GAssistant.IntentAnalyzer.processUserQuery',

'System.AI.GeminiAdapter.ask', // من هذا الملف

'System.AI.GeminiAdapter.askJSON',

'System.AI.GeminiAdapter.askForCodeAnalysis',

'GAssistant.Tools.Developer.reviewCode', // الدوال التي تعمل عليها

'GAssistant.Tools.Finance.addRevenue',

// ... وهكذا

];

return { projectDescription, existingFiles, existingFunctions };

}

// اجعل هذه الدالة متاحة للدالة analyzeAndSuggest

1. **تطوير Utils.gs:** ستحتاج إلى الدوال المساعدة Utils.extractCodeBlocks(text, language) و Utils.removeCodeBlocks(text, language) لاستخلاص الكود من استجابات Gemini.

بهذه التعديلات، GeminiAdapter لن يكون مجرد وسيط، بل سيصبح جزءًا لا يتجزأ من "ذكاء" المساعد، قادرًا على فهم السياق المعقد لطلبات المبرمجين وتقديم حلول متكاملة ومستبقة. هذا هو "الجنون" الذي سيغنيك عن الكثير!

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

### **الخطوات التالية لتفعيل هذا "الجنون":**

**إنشاء AI.ToolExecutor (ملف جديد مثل 20\_ai/8\_ai\_toolExecutor.gs):** هذه الوحدة ستكون حاسمة لتنفيذ الأدوات التي يقترحها Gemini. يجب أن تحتوي على دالة مثل executeFunctionCall(functionName, args) التي يمكنها البحث عن الدوال المسجلة (مثلاً، في DocsManager أو قائمة معرفة مسبقًا) وتنفيذها ديناميكيًا.  
JavaScript  
// مثال مبسط جداً لـ 20\_ai/8\_ai\_toolExecutor.gs

defineModule('System.AI.ToolExecutor', ({ Utils, DocsManager, Telemetry }) => {

function executeFunctionCall(functionName, args) {

Utils.log(`ToolExecutor: Attempting to execute: ${functionName} with args: ${JSON.stringify(args)}`);

Telemetry.track('ToolExecution', { functionName, args });

// مثال بسيط جداً: يجب أن تكون لديك آلية تسجيل للدوال

// هذه مجرد أمثلة، ستحتاج إلى ربطها بدوالك الفعلية

switch (functionName) {

case 'createSheet':

// افترض وجود دالة في Utils أو SpreadsheetTools

return GAssistant.Tools.Spreadsheet.createSheet(args.sheetName);

case 'getFinancialReport':

return GAssistant.Tools.Finance.generateReport(args.reportType, args.period);

case 'sendEmail':

return GAssistant.Tools.Communication.sendEmail(args.to, args.subject, args.body);

// ... أضف المزيد من الدوال هنا

default:

throw new Error(`Function "${functionName}" not found or not executable by ToolExecutor.`);

}

}

return { executeFunctionCall };

});

1. **تحديث AI.Core.ask (في 20\_ai/5\_ai\_core.gs):** يجب أن تكون AI.Core.ask قادرة على معالجة الـ messages array (لـ multi-turn conversation) بدلاً من مجرد prompt string. كما يجب أن تكون قادرة على تمرير tools و toolConfig و thinkingConfig بشكل صحيح إلى Gemini API.
2. **إعداد Config:** تأكد من وجود المتغيرات التالية في ملف 01\_config.gs أو في خصائص المشروع (Project Properties) في Apps Script:
   * GEMINI\_SERVICE\_VERSION
   * GEMINI\_SERVICE\_CACHE\_TTL
   * GEMINI\_SERVICE\_MAX\_RETRIES
   * GEMINI\_SERVICE\_INITIAL\_BACKOFF\_MS
   * GEMINI\_CONVERSATION\_TTL
   * GENERATION\_CONFIG (مثل { temperature: 0.7, maxOutputTokens: 2000 })
   * SAFETY\_SETTINGS (مثل [{ category: 'HARM\_CATEGORY\_DANGEROUS\_CONTENT', threshold: 'BLOCK\_NONE' }])
   * TOOL\_CONFIG (إذا أردت إعدادات أدوات افتراضية عالمية).
3. **تطوير Utils.extractCodeBlocks و Utils.removeCodeBlocks:** إذا لم تكن موجودة بعد في 02\_utils.gs.

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

لتعديلات على الجدول التنفيذي (مُحدث الآن)

| **🔢 رقم** | **📁 اسم الملف / الوحدة** | **📌 المرحلة / المراحل** |
| --- | --- | --- |
|  |  | ... |
| 39 | 30\_tools/6\_image\_processor.gs | 21 |

🔢 **أصبح عدد الملفات الإجمالي: 39 وحدة**

مراحل تطوير الأسس الجديدة

1. تهيئة الوحدات باستخدام defineModule
2. تطبيق نمط Injector لتوزيع التبعيات
3. إعادة ترتيب التشغيل والتهيئة في initializer
4. فصل التبعيات داخل الوظائف
5. تكامل خدمات Google الداخلية
6. بناء دستور الذكاء وإدارة السياق
7. تفعيل Gemini SDK وتهيئة الاتصال
8. هندسة أدوات المحاسبة الذكية
9. ربط أدوات الذكاء مع Tool Calling
10. تنفيذ الوظائف الأساسية: إنشاء، تحرير، بحث
11. تفعيل واجهة التغذية الراجعة الذكية
12. تمكين Gemini من تعديل الكود برمجيًا
13. واجهة التشغيل للمستخدم النهائي
14. دليل الاستخدام الفعلي للمستخدمين
15. تجهيز ملفات العرض للمبرمجين والمستثمرين
16. ربط واجهة المطور مع أدوات التطوير وGemini
17. بناء جسر تحليلي بين Sidebar وذكاء المساعد
18. تمكين Gemini من التحليل السياقي العميق
19. إنشاء زر "تحليل سياقي متقدم" ووحدة ProjectContextTracker
20. أرشفة الملفات وربطها بخطة التنفيذ

المعايير الأساسية لإنشاء كل ملف في المشروع

1. **المعيار المعماري**

* يكون الملف وحدة مستقلة تستخدم defineModule(name, builder)
* لا يعتمد على متغيرات خارجية مباشرة (مثل: كائنات عشوائية أو مكتبات غير مستوردة بشكل صريح)
* يحتوي فقط على ما يخص وظيفته (مبدأ Separation of Concerns)

2. **معيار التسمية والتنظيم**

* اسم الملف يبدأ برقم المجموعة (10\_ui/, 30\_tools/, 20\_ai/) لتسهيل الفرز
* الوحدات تتبع التسمية نوع\_الوحدة/وظيفتها.gs مثل tools\_developer.gs

3. **معيار التوثيق الداخلي**

* كل دالة تُوثّق باستخدام JSDoc
* الوحدات توضّح في البداية هدفها ومجال استخدامها
* تضمين قسم توضيحي لـ @module و @author

4. **معيار السلامة والتشخيص**

* استخدام Utils.log() أو Telemetry.log() لتسجيل الحركات
* حماية الدوال الحساسة باستخدام فحص السياق (if (!authorized) return;)
* تضمين اختبارات داخل tests.gs عند الإمكان

5. **معيار قابلية التوسع**

* لا توجد متغيرات ثابتة داخل الدوال — كلها تأتي من Config.get()
* تسمية دوال قابلة لإعادة الاستخدام (لا تربطها بسياق ضيق مثل forSheetX فقط)
* تفعيل DEBUG\_MODE من خلال Config.gs وليس تعديل الكود يدويًا

6. **معيار التكامل السياقي**

* الدالة تُعلن نفسها داخل Tools.Catalog أو AI.Dispatcher لتكون قابلة للاستدعاء
* تتفاعل مع AI.Context أو ProjectContextTracker عند الحاجة للفهم السياقي

7. **معيار عدم التكرار**

* لا يُعرّف في الملف أي دالة موجودة مسبقًا في Utils أو Config
* التحقق قبل إنشاء دالة مشابهة لكود موجود (خاصة لو فيه process, log, create)

8. **معيار الفاعلية في التحسين**

* الملف يُدمج مع DocsManager لمراقبة تطوره
* يوضح إذا كان قابل للدمج مع ملفات أخرى أو قابل للفصل عند الحاجة

أكيد يا عبدالعزيز 👨‍💻  
 إليك قالب إنشاء ملف معياري في مشروعك — يراعي معايير defineModule, التوثيق الذكي، الربط السياقي، واستدعاء الأدوات داخل الكتالوج.

## **🧩 قالب إنشاء ملف معياري (Google Apps Script)**

/\*\*

\* @module Tools.ExampleModule

\* @description وحدة متخصصة في [اسم الوظيفة أو المجال]

\* @author عبدالعزيز

\*/

defineModule('Tools.ExampleModule', ({ Config, Utils }) => {

/\*\*

\* @function doSomethingUseful

\* @description تنفذ مهمة محددة بناء على البيانات المقدمة

\* @param {Object} input - الكائن الذي يحتوي على المعطيات

\* @returns {Object} - النتيجة بعد المعالجة

\*/

function doSomethingUseful(input) {

Utils.log('تشغيل doSomethingUseful...', input);

// تحقق من الإعدادات

const isEnabled = Config.get('FEATURE\_FLAG\_ENABLED');

if (!isEnabled) return { error: 'الميزة غير مفعلة' };

// تنفيذ المهمة

const result = { output: `ناتج المعالجة لـ ${input.name}` };

return result;

}

/\*\*

\* @function registerInCatalog

\* @description تفعيل الدالة داخل كتالوج الأدوات لجعلها قابلة للاستدعاء من Gemini

\*/

function registerInCatalog() {

GAssistant.ToolsCatalog.registerTool({

id: 'EXAMPLE\_TOOL',

title: '🧪 أداة التجربة',

description: 'تنفذ معالجة تجريبية على مدخلات محددة',

args: [

{ name: 'name', type: 'string', required: true },

{ name: 'value', type: 'number' }

],

handler: doSomethingUseful

});

}

return {

doSomethingUseful,

registerInCatalog

};

});

## **📌 تعليمات عند استخدام القالب**

* ✏️ غيّر ExampleModule باسم الوحدة الفعلي (مثل SheetsProcessor أو GeminiHelper)
* 🎛️ عرّف الدوال بوضوح باستخدام @function, @description, @param, @returns
* 🧠 أضف registerInCatalog() إذا كانت الوحدة ستُستدعى من الشريط الجانبي أو Gemini
* 🔐 استدعِ Config و Utils فقط داخل الدوال، لا تستخدمهم في أعلى الملف خارج defineModule

—-=================================================================

أكيد يا عبدالعزيز، إليك القائمة الشاملة لكل الملفات التي يحتاجها المشروع حسب الهيكل المعماري الحالي، تشمل القديمة، المحدثة، والجديدة، موزعة حسب الوظيفة داخل النظام:

## **📁 الإعدادات والتهيئة**

* config.gs
* utils.gs
* initializer.gs
* security.gs

## **🧠 الذكاء الاصطناعي وتكامل Gemini**

* ai\_constitution.gs
* ai\_core.gs
* ai\_context.gs
* ai\_memory.gs
* ai\_dispatcher.gs
* geminiService.gs
* GeminiAdapter.gs
* AI.Bookkeeper.gs

## **🧰 أدوات النظام**

* tools\_catalog.gs
* tools\_accounting.gs
* tools\_sheets.gs
* tools\_codeReview.gs
* tools\_developer.gs

## **🧑‍💻 وكلاء التشغيل والمساعدين**

* agents\_cfo.gs
* agents\_developer.gs
* agents\_triggers.gs

## **🎛️ واجهة المستخدم والتفاعل**

* ui.gs
* ui\_dialogue.gs
* ui\_developerSidebar.gs
* DeveloperSidebar.html
* AdvancedDevPanel.html

## **📡 التشغيل والدعم التقني**

* api.gs
* code.gs
* tests.gs
* telemetry.gs

## **🧭 التحليل والتوجيه الذكي**

* intentAnalyzer.gs
* ProjectContextTracker.gs

## **🧾 التوثيق والعرض والتشغيل الخارجي**

* DocsManager.gs
* DeveloperWorkshop.gs
* DocumentationIndex.gs

## **📄 ملفات التوثيق والعرض الخارجي**

* Readme.md
* UserUsageGuide.md
* DeveloperOverview.md
* InvestorPitch.md

## **🔢 المجموع الكامل**

**عدد الملفات المطلوبة = 38 ملفًا ووحدة**

هل ترغب أن أجهز جدول فيه ✅ حالة كل ملف، 📎 المرحلة المرتبطة به، و📁 نوعه؟  
 أو ورقة Google Sheet فيها قائمة التشغيل لكل وحدة؟  
 مستعد نبدأ بتنفيذها وحدة وحدة — حسب الأولوية والارتباط 💼📦✨