**استخدام Gemini AI لتطوير وتحسين الأكواد في Google Sheets عبر Google Apps Script**

**1. مقدمة: قوة الذكاء الاصطناعي في Google Apps Script**

**تُعد Google Apps Script (GAS) بيئة تطوير سريعة قائمة على السحابة، تمكّن المستخدمين من أتمتة المهام وتوسيع وظائف تطبيقات Google Workspace، ولا سيما Google Sheets. تعتمد المنصة على لغة JavaScript الحديثة، وتوفر مكتبات مدمجة للتفاعل السلس مع خدمات Google مثل Gmail وCalendar وDrive. يمكن للمطورين من خلالها إنشاء دوال مخصصة (Custom Functions) تُستدعى مباشرةً من خلايا Google Sheets، وتطوير وحدات ماكرو، وبناء تطبيقات ويب متكاملة. من أبرز مزايا Apps Script أنه لا يتطلب أي تثبيت محلي؛ حيث يتوفر محرر أكواد مباشر داخل المتصفح، وتُحفظ السكريبتات وتُشغل بالكامل على خوادم Google، مما يسهل الوصول إليها وإدارتها.**

**في المقابل، تُمثل Gemini AI عائلة من نماذج اللغة الكبيرة متعددة الوسائط (LLMs) التي طورتها Google DeepMind. تتميز هذه النماذج بقدرتها على معالجة المدخلات النصية والصوتية والمرئية، وتوليد مخرجات نصية وصورية. تتوفر نماذج متقدمة مثل Gemini 2.5 Flash، الذي يوفر سرعة وكفاءة عالية للمهام المتكررة، وGemini 2.5 Pro، الذي يقدم قدرات تفكير معقدة. لتعزيز تجربة المطورين، تقدم Google أدوات مساعدة مثل Gemini Code Assist، التي توفر مساعدة برمجية مدعومة بالذكاء الاصطناعي. تشمل هذه المساعدة إكمال الكود، وتوليد الدوال، واختبارات الوحدة، والمساعدة في تصحيح الأخطاء، وتوثيق الكود ضمن بيئات التطوير المتكاملة (IDEs) المدعومة مثل VS Code وJetBrains IDEs.**

**يركز هذا التقرير على كيفية دمج والاستفادة من قدرات Gemini AI، سواء بشكل مباشر أو غير مباشر عبر Google Apps Script، لتعزيز دورة حياة تطوير الأكواد داخل بيئة Google Sheets. سيغطي التقرير الجوانب العملية لتوليد الأكواد الجديدة (الوحدات، السكريبتات، الدوال)، وتحديد وتصحيح الأخطاء البرمجية، وتقديم اقتراحات لتحسين جودة الأكواد والوحدات والسكريبتات والدوال الخاصة بـ Apps Script.**

**يبرز هذا التقارب بين Apps Script وGemini AI ضرورة سد الفجوة بين قدرات Apps Script الأصلية وإمكانيات الذكاء الاصطناعي المتقدمة. فبينما يُعد Apps Script أداة قوية للأتمتة داخل Google Workspace ، يتكامل Gemini AI بشكل أساسي مع بيئات تطوير متكاملة محددة. هذا يستدعي البحث عن طرق لجعل مساعدة الذكاء الاصطناعي تبدو متكاملة مع تجربة تطوير Apps Script، حتى لو لم يكن هناك تكامل مباشر مع محرر Apps Script. يتطلب هذا غالبًا استخدام أغلفة مخصصة أو مكتبات مساعدة، وهو ما سيُفصل في الأقسام اللاحقة.**

**علاوة على ذلك، يُسهم هذا التكامل في إضفاء الطابع الديمقراطي على البرمجة المتقدمة لمستخدمي Google Sheets المتقدمين. يُصنف Apps Script كـ "منصة منخفضة الكود" تُمكّن "مستخدمي الأعمال من بناء حلول مخصصة... دون الحاجة إلى خبرة تطوير احترافية". يُسهم Gemini AI، من خلال تقديم توليد الكود وتصحيح الأخطاء ، في خفض حاجز الدخول للبرمجة المعقدة بشكل كبير. هذا يعني أن المستخدمين الذين يتمتعون بالكفاءة في Sheets ولكن ليسوا بالضرورة مبرمجين خبراء يمكنهم الآن الاستفادة من الذكاء الاصطناعي لإنشاء أتمتة ووظائف أكثر تعقيدًا. تُشير هذه القدرة إلى تحول في كيفية تفاعل المستخدمين المتقدمين مع Google Sheets، والانتقال نحو استخدام السكريبتات المخصصة بمساعدة الذكاء الاصطناعي.**

**2. دمج Gemini AI مع Google Apps Script**

**يتطلب دمج Gemini AI مع Google Apps Script إعدادًا دقيقًا لتمكين التواصل بين البيئتين. تبدأ العملية بإنشاء مشروع Apps Script جديد، والذي يمكن تحقيقه بسهولة بالانتقال إلى script.new في المتصفح. يُنصح بتسمية المشروع بشكل وصفي، مثل "AI Studio"، لتسهيل إدارته.**

**الخطوة الأساسية التالية هي إعداد مفتاح API الخاص بـ Gemini. لا ينبغي تضمين هذا المفتاح مباشرة في الكود لضمان الأمان. بدلاً من ذلك، يُخزن المفتاح بأمان في إعدادات المشروع ضمن "خصائص السكريبت" (Script Properties). يتم ذلك بإضافة خاصية جديدة باسم GEMINI\_API\_KEY وإدخال قيمة مفتاح API الخاص بك. يُعد هذا الأسلوب ضروريًا لحماية المعلومات الحساسة. بالنسبة لبيئات الإنتاج أو الاستخدام المتقدم، يُنصح بربط مشروع Apps Script بمشروع Google Cloud Platform (GCP) وتمكين Vertex AI API. يتطلب هذا التكامل إضافة نطاقات OAuth محددة إلى ملف**

**appsscript.json الخاص بالمشروع، مثل https://www.googleapis.com/auth/cloud-platform و https://www.googleapis.com/auth/script.external\_request، للسماح للسكريبت بالوصول إلى خدمات Google السحابية وإجراء طلبات HTTP خارجية.**

**بعد إعداد المشروع ومفتاح API، يمكن لسكريبتات Apps Script التفاعل مع Gemini API. يتم ذلك عادةً باستخدام خدمة UrlFetchApp المدمجة في Apps Script لإرسال طلبات POST إلى نقطة نهاية generateContent الخاصة بـ Gemini. يجب أن تتضمن حمولة (payload) الطلب كائن JSON يحدد المحتوى المراد معالجته (مثل النص أو الكود) ونموذج Gemini المراد استخدامه، مثل**

**gemini-2.5-flash أو gemini-1.5-pro. من الضروري تضمين مفتاح API في رؤوس الطلب (headers) كـ**

**x-goog-api-key لضمان المصادقة الصحيحة للطلب.**

**على الرغم من إمكانية إجراء استدعاءات UrlFetchApp المباشرة، والتي توفر أقصى قدر من التحكم وأقل قدر من التبعيات، إلا أنها غالبًا ما تكون مطولة وتتطلب تحليل JSON يدويًا. لتبسيط عملية التفاعل مع Gemini API من Apps Script، يمكن للمطورين الاستفادة من المكتبات المساعدة المخصصة مثل GenAIApp. توفر هذه المكتبات واجهة برمجة تطبيقات (API) أكثر سهولة لإدارة المحادثات متعددة الأدوار، وإضافة أنواع مختلفة من الرسائل (نص، صور، ملفات)، وتعريف الدوال القابلة للاستدعاء بواسطة الذكاء الاصطناعي (Function Calling)، وتمكين ميزات مثل تصفح الويب والوصول المباشر إلى Google Sheets. تسمح**

**GenAIApp بتحديد مفتاح API أو المصادقة عبر Google Cloud، وإنشاء مثيلات Chat جديدة، وإضافة رسائل المستخدم والنظام، وتحديد FunctionObjects التي يمكن لـ Gemini استدعاؤها لتنفيذ إجراءات محددة داخل بيئة Apps Script. بالنسبة لمعظم مطوري Apps Script، ستكون المكتبة التي يتم صيانتها جيدًا مفضلة لتحقيق الكفاءة في عملية التطوير وتقليل الكود المتكرر.**

**تُعد إدارة مفاتيح API بشكل آمن أمرًا بالغ الأهمية. بينما يمكن إجراء استدعاءات API مباشرة، فإن تخزين مفاتيح API مباشرة في الكود يمثل خطرًا أمنيًا كبيرًا. يُذكر PropertiesService صراحةً كطريقة لتخزين مفاتيح API بشكل آمن. يوضح**

**PropertiesService كذلك كيفية تخزين البيانات الحساسة باستخدام خصائص السكريبت أو المستخدم أو المستند، مؤكدًا أن "الخصائص لا تتم مشاركتها أبدًا بين السكريبتات". هذا يسلط الضوء على أفضل ممارسة أمنية حاسمة: يجب على المطورين استخدام**

**PropertiesService لمفاتيح API وفهم النطاقات المختلفة (السكريبت، المستخدم، المستند) لمنع التعرض غير المقصود، خاصة بالنسبة للإضافات أو السكريبتات المشتركة. هذا يضمن أن عملية التكامل لا تُسهل فقط الوظائف المتقدمة ولكنها تحافظ أيضًا على معايير أمنية عالية.**

**3. Gemini AI لتوليد وتطوير الأكواد**

**يمتلك Gemini AI قدرة كبيرة على توليد كود برمجي كامل أو كتل كود بناءً على الأوصاف باللغة الطبيعية. يمكن للمطورين تزويد Gemini بموجهات (prompts) تصف الوظيفة المطلوبة، مثل "إنشاء دالة لجمع البيانات من نطاق معين في Google Sheet" أو "اكتب سكريبت Apps Script لأتمتة إرسال رسائل البريد الإلكتروني". على الرغم من أن Gemini Code Assist لا يتكامل مباشرة مع محرر Apps Script في الوقت الحالي ، مما يعني أن المطورين يحتاجون إلى نسخ الكود المولّد ولصقه يدويًا في ملفات**

**.gs الخاصة بمشروع Apps Script.**

**تسمح ميزة استدعاء الدوال (Function Calling) لنموذج Gemini بتحليل طلب المستخدم وتحديد ما إذا كان استدعاء دالة محددة سيكون مفيدًا لتلبية الطلب. يمكن للمطورين تعريف "إعلانات الدوال" (Function Declarations) في كود التطبيق الخاص بهم. تصف هذه الإعلانات اسم الدالة ومعاملاتها والغرض منها لنموذج Gemini، مما يسمح له بفهم السياق واستدعاء الدالة المناسبة. عندما يحدد Gemini الحاجة إلى دالة، فإنه يستجيب بكائن JSON منظم يحتوي على اسم الدالة والوسائط المطلوبة لتشغيلها. يقع على عاتق التطبيق (Apps Script في هذه الحالة) تنفيذ الكود المقابل للدالة باستخدام الوسائط المقدمة من Gemini، ثم استخدام النتيجة لتوليد استجابة نهائية سهلة الاستخدام للمستخدم. يُعد هذا النهج مفيدًا بشكل خاص لأتمتة Apps Script الخاصة بالمجال، حيث يمكن للمطورين عرض وظائف Apps Script (مثل**

**updateSheetData أو sendEmailReport) على Gemini كأدوات قابلة للاستدعاء. هذا يسمح بسير عمل أتمتة عالي التخصيص وذكي، حيث يمكن لـ Gemini "تقرير" استدعاء وظيفة Apps Script بناءً على موجه طبيعي للغة المستخدم، مما يؤدي بشكل فعال إلى إنشاء طبقة أتمتة أكثر ذكاءً فوق Google Workspace.**

**تُقدم Gemini AI أمثلة عملية متعددة للمهام الخاصة بـ Google Sheets:**

**توليد الدوال المخصصة (Custom Functions): يمكن لـ Gemini توليد دوال JavaScript يمكن استخدامها مباشرة كدوال مخصصة في Google Sheets. على سبيل المثال، يمكن للمطور أن يطلب من Gemini إنشاء دالة لحساب الربح (**

**calculateProfit) أو دالة =AI() جديدة في Sheets لأتمتة مهام معالجة النصوص أو التصنيف داخل الخلايا.**

**معالجة البيانات وأتمتتها: يمكن لـ Gemini المساعدة في إنشاء سكريبتات Apps Script لأتمتة مهام معالجة البيانات المعقدة في Sheets، مثل تحليل البيانات لتحديد الاتجاهات، أو تحديث التقارير تلقائيًا بناءً على بيانات جديدة، أو استخراج البيانات من رسائل البريد الإلكتروني والمستندات.**

**توليد الصيغ والجداول والرسوم البيانية: يمكن لـ Gemini المساعدة في توليد صيغ Google Sheets المعقدة، وإنشاء جداول منظمة بناءً على وصف نصي، وتوليد رسوم بيانية بناءً على البيانات الموجودة في ورقة العمل، مما يسهل التحليل والتصور للمستخدمين غير المبرمجين.**

**من المهم الإشارة إلى أن الكود المولّد بواسطة الذكاء الاصطناعي يُعد نقطة انطلاق، وليس حلاً نهائيًا. تُشير Google إلى أن "منتجات Gemini لـ Google Cloud يمكن أن تولد مخرجات تبدو معقولة ولكنها غير صحيحة من الناحية الواقعية. نوصي بالتحقق من صحة جميع المخرجات من منتجات Gemini لـ Google Cloud قبل استخدامها". هذا يعني أن العنصر البشري في الحلقة لا يزال حاسمًا؛ يجب على المطورين مراجعة الكود واختباره وربما تحسينه. تُصبح المراجعة اليدوية والإدخال ضرورة، خاصة في سياق Apps Script حيث يكون التكامل المباشر محدودًا حاليًا.**

**4. Gemini AI لتحديد وتصحيح الأخطاء البرمجية (Debugging)**

**يواجه مطورو Apps Script أنواعًا مختلفة من الأخطاء التي يمكن لـ Gemini AI المساعدة في تشخيصها وحلها. تتضمن الأخطاء الشائعة في Apps Script أخطاء بناء الجملة (Syntax errors)، والتي تُكتشف فور محاولة حفظ السكريبت، مثل الأقواس المفقودة أو الأخطاء الإملائية. أما أخطاء وقت التشغيل (Runtime errors)، فتظهر أثناء تنفيذ السكريبت، وقد تنجم عن استخدام دالة أو فئة بشكل غير صحيح، أو تجاوز حصص الاستخدام اليومية لخدمات Google. بالإضافة إلى ذلك، يمكن أن تنشأ أخطاء Gemini API نفسها بسبب تجاوز حدود المعدل (rate limits)، أو استخدام مفتاح API غير صحيح أو غير مصرح به، أو طلبات غير صالحة (malformed requests)، أو مشاكل في الاتصال بالخدمة أو عدم توفرها مؤقتًا.**

**يمكن لـ Gemini Code Assist المساعدة بشكل كبير في تصحيح الأخطاء من خلال تحليل الكود، وتحديد المشكلات المحتملة، واقتراح خطوات استكشاف الأخطاء وإصلاحها. يمكن للمطورين تزويد Gemini بمقتطفات الكود أو رسائل الخطأ التي يواجهونها، وطلب "مساعدتي في تصحيح الكود الخاص بي" أو "شرح هذا الكود لي". يوفر Gemini تفسيرات واضحة للأخطاء ويقترح حلولاً، كما هو موضح في أمثلة تصحيح أخطاء JavaScript في Chrome DevTools، حيث يمكنه تحليل أخطاء Console وتقديم إصلاحات مقترحة. يمكن لـ Gemini أيضًا تحليل شجرة استدعاء الأداء (flame graph) لتحديد مشكلات الأداء في الكود واقتراح تحسينات.**

**تُعد القدرة على استخدام Gemini لتحليل الكود بشكل استباقي ذات قيمة كبيرة. فبدلاً من انتظار أخطاء وقت التشغيل، يمكن للمطورين استخدام Gemini لتحليل الكود قبل النشر. تُشير القدرة على "تحليل الكود الموجود، وتحديد المشكلات المحتملة، واقتراح التحسينات" إلى أن Gemini يمكن أن يعمل كأداة لتحليل الكود الثابت. من خلال تزويد Gemini بمقتطفات كود Apps Script مع موجهات مثل "حلل كود Apps Script التالي بحثًا عن الأخطاء المحتملة واقترح تحسينات" ، يمكن للمطورين اكتشاف المشكلات قبل النشر، مما يقلل من وقت تصحيح الأخطاء ويُحول تصحيح الأخطاء من رد فعل إلى استباقي.**

**عند التعامل مع أخطاء Gemini API المحددة، يُعد فهم رموز الأخطاء الشائعة أمرًا حيويًا. يوضح الجدول 1 رموز أخطاء Gemini API الشائعة وحلولها:**

**الجدول 1: رموز أخطاء Gemini API الشائعة وحلولها**

**رمز HTTP**

**الحالة**

**الوصف**

**الحل**

**400**

**INVALID\_ARGUMENT**

**جسم الطلب مشوه أو حقل مطلوب مفقود.**

**التحقق من تنسيق الطلب والأمثلة والإصدارات المدعومة.**

**403**

**PERMISSION\_DENIED**

**مفتاح API لا يملك الأذونات المطلوبة.**

**التأكد من تعيين مفتاح API بشكل صحيح ولديه الوصول المناسب.**

**429**

**RESOURCE\_EXHAUSTED**

**تجاوز حد المعدل لعدد الطلبات في الدقيقة.**

**التأكد من البقاء ضمن حدود النموذج أو طلب زيادة الحصة.**

**500**

**INTERNAL**

**خطأ غير متوقع حدث من جانب Google.**

**تقليل سياق الإدخال، أو التبديل مؤقتًا إلى نموذج آخر، أو الانتظار وإعادة المحاولة.**

**503**

**UNAVAILABLE**

**الخدمة مثقلة مؤقتًا أو غير متاحة.**

**التبديل مؤقتًا إلى نموذج آخر، أو الانتظار وإعادة المحاولة.**

**504**

**DEADLINE\_EXCEEDED**

**الخدمة غير قادرة على إنهاء المعالجة ضمن المهلة المحددة (غالباً بسبب حجم الموجه).**

**تعيين "مهلة" أكبر في طلب العميل.**

**تُشكل قيود نافذة السياق تحديًا وفرصة في نفس الوقت. نماذج Gemini تدعم طول سياق كبير يصل إلى 32 ألف رمز، ولكن تحليل المشاريع الكبيرة جدًا قد يتجاوز هذه الحدود. لمواجهة ذلك، يمكن استخدام استراتيجيات مثل تقسيم الكود أو البيانات إلى أجزاء (chunking) ومعالجة كل جزء على حدة، ثم تجميع النتائج. هذا يشير إلى استراتيجية لتصحيح أخطاء مشاريع Apps Script الكبيرة: قد يحتاج المطورون إلى تنفيذ آلية تقسيم مماثلة أو استخدام أدوات مثل "Context-Packer" لتزويد Gemini بسياق ملخص وذو صلة لتصحيح الأخطاء، بدلاً من المشروع بأكمله. هذا يحول القيد إلى فرصة لإدارة سياق أكثر ذكاءً.**

**5. Gemini AI لتحسين وتطوير الأكواد**

**تُسهم Gemini AI بشكل كبير في تحسين جودة الأكواد من خلال تقديم اقتراحات لزيادة قابلية القراءة والصيانة. يمكن للنموذج المساعدة في تحسين قابلية قراءة الكود وصيانته عن طريق اقتراح تحسينات في التنسيق، وإضافة تعليقات توضيحية، وإعادة تسمية المتغيرات والدوال لزيادة الوضوح. يمكن للمطورين توجيه Gemini بموجهات مثل "اجعل الكود المحدد أكثر قابلية للقراءة" أو "لخص ما تفعله هذه الدالة".**

**بالإضافة إلى ذلك، يمكن لـ Gemini تحليل الكود الحالي واقتراح تحسينات هيكلية، تُعرف بإعادة الهيكلة (Refactoring). تشمل هذه التحسينات إزالة الكود الميت (dead code)، واتباع الأنماط المعروفة، وتعزيز المعيارية (modularity)، وفصل الاهتمامات (separation of concerns)، وقابلية التوسع، والصيانة. يمكنه أيضًا المساعدة في تحديد المشكلات المحتملة في الكود الحالي واقتراح تحسينات. ومع ذلك، يجب على المطورين أن يكونوا على دراية بأن الكود المولّد بالذكاء الاصطناعي قد يحتوي على ثغرات أمنية أو لا يلتزم دائمًا بأفضل الممارسات الأمنية ، مما يستدعي المراجعة الدقيقة.**

**تتجاوز قدرات Gemini مجرد توليد الكود؛ يمكنه "إعادة هيكلة" و "تصحيح أخطاء" و "شرح" و "جعل الكود أكثر قابلية للقراءة". هذا يضع Gemini كـ "مبرمج مساعد" مدعوم بالذكاء الاصطناعي لمطوري Apps Script. يمكنه تقديم اقتراحات في الوقت الفعلي ، وتحليل الكود الموجود، وتحديد المشكلات. هذا يشير إلى سير عمل تعاوني حيث يعزز الذكاء الاصطناعي قدرات المطور، بدلاً من استبدالها، مما يؤدي إلى كود Apps Script عالي الجودة وأكثر قابلية للصيانة.**

**يستطيع Gemini Code Assist أيضًا أتمتة توليد اختبارات الوحدة (Unit Tests) للكود المحدد، مما يساهم في تحسين جودة الكود وتقليل الأخطاء من خلال توفير تغطية اختبارية تلقائية. يمكن للمطورين تزويد Gemini بالكود المصدر والكود الاختباري الموجود (إن وجد) لطلب ملاحظات حول الحالات الهامشية المفقودة (missing edge cases)، وجودة الاختبارات، ومدى قابلية الكود للاختبار.**

**على الرغم من أن الذكاء الاصطناعي يمكنه اقتراح تحسينات وتوليد اختبارات، تُحذر الوثائق مرارًا وتكرارًا من احتمال وجود "مخرجات غير صحيحة من الناحية الواقعية" أو "كود غير موثوق به" و "ثغرات أمنية". هذا يُبرز أن الذكاء الاصطناعي يُسرع تحسين الكود، ولكنه في الوقت نفسه يزيد من الحاجة إلى التحقق البشري. قد يؤدي الاعتماد المفرط على الذكاء الاصطناعي إلى "ديون تقنية" أو "فقدان تحكم المطور". لذلك، تظل "الرقابة المناسبة" و "الاختبار" و "مراجعات الكود" ذات أهمية قصوى لضمان جودة الكود وسلامته.**

**6. تكامل Gemini المباشر ضمن Google Sheets**

**يُقدم Gemini تكاملاً مباشرًا وفعالًا داخل Google Sheets، مما يُمكّن المستخدمين من الاستفادة من قدرات الذكاء الاصطناعي دون الحاجة إلى كتابة أكواد Apps Script معقدة. يمكن للمستخدمين التفاعل مباشرة مع Gemini داخل Google Sheets لإنشاء جداول منظمة، وصيغ معقدة، ورسوم بيانية متنوعة.**

**للوصول إلى Gemini في Sheets، يمكن النقر على أيقونة "Ask Gemini" في الزاوية العلوية اليمنى من واجهة Google Sheets. بدلاً من ذلك، يمكن للمستخدمين ببساطة بدء كتابة = في أي خلية ثم استخدام اختصار لوحة المفاتيح (Ctrl + Alt + g لنظامي Windows و Chrome OS، أو ⌘ + Ctrl + g لنظام Mac OS) لتنشيط وظيفة الذكاء الاصطناعي المباشرة. تُعرف هذه الوظيفة أيضًا باسم**

**=AI(). يمكن للمستخدمين تقديم موجهات باللغة الطبيعية مثل "إنشاء جدول لحدث فريق ليوم كامل"، "إنشاء صيغة تقسم الأهداف على المباريات"، أو "إنشاء مخطط يوضح النسبة المئوية للمصروفات على إجمالي الراتب بعد الضريبة".**

**بالإضافة إلى توليد المحتوى، يمكن لـ Gemini تحليل البيانات الموجودة في Sheets، وتحديد الاتجاهات، وتقديم رؤى، وشرح خطوات التحليل التي قام بها. يمكنه أيضًا تعديل ورقة العمل مباشرة بناءً على الموجهات النصية، مثل "تمييز القيم الأقل من 100". تتميز قدرة Gemini في Sheets بالقدرة على الرجوع إلى الملفات من Google Drive أو رسائل البريد الإلكتروني من Gmail لتوليد استجابات ذات سياق غني بالمعلومات، مما يوسع نطاق فائدته إلى ما هو أبعد من بيانات جدول البيانات نفسها.**

**يُشير التكامل المباشر لـ Gemini في Google Sheets لمهام مثل توليد الصيغ والجداول والرسوم البيانية إلى نهج "بدون كود/منخفض الكود" للذكاء الاصطناعي. لا يحتاج المستخدمون إلى كتابة Apps Script أو استدعاء واجهات برمجة التطبيقات؛ يمكنهم التفاعل مع الذكاء الاصطناعي باستخدام اللغة الطبيعية مباشرة داخل واجهة جدول البيانات. هذا يوسع إمكانية الوصول إلى الأتمتة والتحليل المدعومين بالذكاء الاصطناعي بشكل كبير، مما يمكّن غير المطورين من الاستفادة من القدرات المتقدمة دون كتابة سطر واحد من الكود.**

**تُكمل هذه القدرات المباشرة أتمتة Apps Script. فبينما يتعامل التكامل المباشر في Sheets مع العديد من المهام، لا تزال الأتمتة المعقدة متعددة الخطوات أو التكاملات مع الخدمات الخارجية تتطلب Apps Script. يمكن لـ Gemini في Sheets توليد صيغة، ويمكن بعد ذلك استخدام Apps Script لتطبيق هذه الصيغة عبر العديد من أوراق العمل، أو تشغيل إجراءات بناءً على مخرجاتها. أو يمكن لـ Gemini في Sheets تحليل البيانات، ويمكن بعد ذلك استخدام Apps Script لهذه الرؤية لإنشاء تقرير أو إرسال بريد إلكتروني. هذا يخلق تآزرًا قويًا حيث يتعامل الذكاء الاصطناعي المباشر في Sheets مع التحليل المخصص والتوليد البسيط، بينما يتعامل Apps Script مع الأتمتة البرمجية المتطورة والتكامل.**

**7. أفضل الممارسات، القيود، والاعتبارات الأمنية**

**لتحقيق أقصى استفادة من Gemini AI في تطوير Apps Script، تُعد هندسة الأوامر الفعالة (Prompt Engineering) أمرًا بالغ الأهمية للحصول على استجابات دقيقة وعالية الجودة من نماذج اللغة. تتضمن أفضل الممارسات في هذا المجال ما يلي:**

**توفير السياق والتفاصيل: يجب كتابة الموجهات كما لو كنت تتحدث مع شخص، مع تضمين تفاصيل حول سبب إنجاز المهمة ومستوى الخبرة المطلوب (على سبيل المثال، "أنت مطور برامج خبير").**

**تضمين أمثلة (Few-shot prompting): استخدام أمثلة قليلة في الموجه (few-shot prompts) لتوجيه النموذج بشأن التنسيق، والعبارات، والنطاق، أو الأنماط العامة المطلوبة للاستجابة.**

**تقسيم المشكلات المعقدة: يجب تقسيم المهام المعقدة إلى مكونات أبسط أو تسلسل الموجهات (chain prompts) لخطوات متتابعة، حيث يصبح مخرج موجه واحد مدخلًا للموجه التالي.**

**تحديد اللغة والمنتجات: تحديد لغة البرمجة المطلوبة (مثل JavaScript لـ Apps Script) والمنتجات أو التقنيات المحددة.**

**على الرغم من الفوائد الكبيرة، يواجه استخدام الذكاء الاصطناعي في توليد الأكواد تحديات يجب معالجتها بفعالية:**

**موثوقية الكود: يمكن أن يحتوي الكود المولّد بواسطة الذكاء الاصطناعي على أخطاء، عيوب، أو ثغرات أمنية، حتى لو بدا وظيفيًا. تشير الدراسات إلى أن 30-50% من الكود المولّد بالذكاء الاصطناعي يحتوي على ثغرات أمنية.**

**الديون التقنية (Technical Debt): قد يؤدي الاعتماد المفرط على الذكاء الاصطناعي لإصلاحات سريعة أو مهام متكررة إلى كود يصعب صيانته على المدى الطويل، مما قد يبطئ التطوير ويزيد التكاليف بمرور الوقت.**

**قيود نافذة السياق (Context Window Limits): نماذج Gemini تدعم طول سياق كبير (حتى 32 ألف رمز)، ولكن تحليل المشاريع الكبيرة جدًا قد يتجاوز هذه الحدود. لمواجهة ذلك، يمكن استخدام استراتيجيات مثل تقسيم الكود أو البيانات إلى أجزاء (chunking) ومعالجة كل جزء على حدة، ثم تجميع النتائج.**

**فقدان تحكم المطور: قد يقلل الاعتماد المفرط على أدوات الذكاء الاصطناعي من فهم المطور للكود الأساسي، مما يجعل تصحيح الأخطاء، أو التحسين، أو توسيع نطاق الكود أكثر صعوبة.**

**تُعد أفضل الممارسات الأمنية ضرورية عند التعامل مع الأكواد التي يولدها الذكاء الاصطناعي وإدارة مفاتيح API:**

**تخزين مفاتيح API بشكل آمن: يجب تخزين مفاتيح API الحساسة باستخدام PropertiesService في Apps Script، والذي يوفر مخازن منفصلة للخصائص على مستوى السكريبت، والمستخدم، والمستند. هذا يمنع تضمين المفاتيح مباشرة في الكود، مما يقلل من مخاطر التعرض.**

**التحقق والاختبار الدقيق للكود المولّد: يجب على المطورين دائمًا التحقق من صحة الكود المولّد بواسطة الذكاء الاصطناعي واختباره بدقة (بما في ذلك اختبارات الوحدة والتكامل) قبل النشر لضمان خلوه من الأخطاء والثغرات الأمنية.**

**دمج أدوات الأمان: بالنسبة للمشاريع الكبيرة أو الحساسة، يُنصح بدمج اختبار أمان التطبيقات الثابت (SAST)، وتحليل تكوين البرامج (SCA)، والاختبار الديناميكي (DAST) في خطوط التكامل المستمر/النشر المستمر (CI/CD).**

**مراجعة الكود البشري: لا يزال الإشراف البشري ومراجعات الكود ضروريين لتحديد العيوب الأمنية المحتملة، وضمان الامتثال لأفضل الممارسات، والحفاظ على جودة الكود الشاملة.**

**الوعي بالبيانات والخصوصية: يجب أن يكون المطورون على دراية بكيفية استخدام Gemini لبياناتهم. تؤكد Google أن بيانات Workspace الخاصة بالمؤسسة لا تُستخدم لتدريب نماذج Gemini أو لاستهداف الإعلانات، مما يوفر مستوى من الخصوصية.**

**نظرًا لأن الذكاء الاصطناعي لا يزال تقنية ناشئة، فإن المخرجات قد تكون غير صحيحة واقعيًا أو تحتوي على ثغرات أمنية أو أخطاء منطقية. لذلك، يجب على المطورين دائمًا التحقق من صحة جميع المخرجات من منتجات Gemini واختبارها بدقة قبل استخدامها في بيئات الإنتاج. يُعد الاختبار الشامل ضروريًا لضمان موثوقية الكود وأمنه وأدائه الأمثل.**

**يُشير هذا إلى تحدي "التحيز والمساءلة" في الكود المولّد بواسطة الذكاء الاصطناعي. يذكر أن "خوارزميات الذكاء الاصطناعي ليست مسؤولة عن الأخطاء التي تحدث، ولا يمكنها توفير الشفافية حول آلياتها وعملياتها الداخلية". كما يُلاحظ خطر "البرمجة التي تعزز الصور النمطية الضارة أو تروج للمعلومات المضللة.". إذا أدخل الكود المولّد بواسطة الذكاء الاصطناعي تحيزات دقيقة أو أنماطًا غير آمنة ليست واضحة على الفور، فمن المسؤول؟ هذا يُبرز الحاجة إلى اختبار قوي يتجاوز الوظائف، بما في ذلك التدقيق الأمني وربما الكشف عن التحيز، وهو تحدٍ كبير لمطوري Apps Script.**

**يُؤدي دمج الذكاء الاصطناعي إلى تحويل دور المطور من مجرد كتابة الكود إلى إدارة والتحقق من صحة الكود المولّد بواسطة الذكاء الاصطناعي. يتطلب هذا مهارات جديدة: هندسة الموجهات ، فهم قيود الذكاء الاص4ناعي ، ومراجعة تصحيح الأخطاء/الأمان المتقدمة. يُصبح المطور أشبه بـ "منسق الذكاء الاصطناعي" و "أخصائي ضمان الجودة" لسير العمل المدعوم بالذكاء الاصطناعي.**

**الجدول 2: اعتبارات رئيسية للأكواد التي يولدها الذكاء الاصطناعي**

**فئة المخاطر**

**الوصف**

**استراتيجية التخفيف**

**موثوقية الكود**

**قد يحتوي الكود المولّد على أخطاء أو عيوب منطقية.**

**الاختبار الشامل (الوحدة، التكامل، النظام).**

**الثغرات الأمنية**

**إدخال غير مقصود لنقاط ضعف أمنية.**

**مراجعة الكود البشري الدقيقة، دمج أدوات SAST/SCA/DAST.**

**الديون التقنية**

**كود يعمل حالياً ولكنه صعب الصيانة أو التوسع مستقبلاً.**

**التصميم المعياري، الالتزام بأفضل الممارسات، عدم الاعتماد المفرط على الذكاء الاصطناعي.**

**فقدان تحكم المطور**

**تقليل فهم المطور للكود الأساسي بسبب الاعتماد المفرط.**

**المراجعة النشطة للكود، التعلم المستمر، الفهم العميق للوظائف المولّدة.**

**قيود نافذة السياق**

**صعوبة معالجة المشاريع الكبيرة جدًا في طلب واحد.**

**تقسيم الكود/البيانات إلى أجزاء، استخدام أدوات إدارة السياق.**

**8. الخاتمة: مستقبل تطوير Apps Script بمساعدة الذكاء الاصطناعي**

**يُعزز Gemini AI بشكل كبير إنتاجية المطورين عن طريق أتمتة المهام المتكررة، وتوليد الكود الأساسي، وتسريع عمليات تصحيح الأخطاء، وتسهيل الاتساق في الكود عبر الفرق. يقلل هذا من العبء الذهني على المطورين، مما يسمح لهم بالتركيز على المهام الأكثر تعقيدًا وحل المشكلات الإبداعي. الأهم من ذلك، أنه يُمكّن المستخدمين غير المتخصصين في البرمجة من إنشاء حلول مخصصة وأتمتة المهام المعقدة داخل Google Workspace، مما يوسع نطاق استخدام Apps Script بشكل كبير.**

**تتطور قدرات Gemini باستمرار، مع التركيز على تحسين التفكير، وتوليد المحتوى متعدد الوسائط، وقدرات تنفيذ الكود. يشير الدمج الأعمق للذكاء الاصطناعي في Google Workspace، كما يتضح من ميزات Gemini المدمجة في Docs وSheets ، إلى مستقبل حيث يصبح الذكاء الاصطناعي مساعدًا دائمًا ومتكاملًا عبر جميع تطبيقات Google. توفر القدرة على تخصيص استجابات الذكاء الاصطناعي بناءً على مستودعات الكود الخاصة بالمؤسسة (كما هو الحال مع Gemini Code Assist Enterprise) مسارات متقدمة لتطوير التطبيقات التي تتوافق مع أفضل الممارسات التنظيمية.**

**من المرجح أن يتضمن المستقبل بيئة تطوير "مدعومة بالذكاء الاصطناعي" بشكل أكبر داخل محرر Apps Script نفسه. بينما ينص على عدم وجود تكامل مباشر مع Gemini Code Assist حاليًا ، فإن اتجاه تكامل الذكاء الاصطناعي في بيئات تطوير Google Cloud الأخرى وميزات الذكاء الاصطناعي المباشرة في Sheets وDocs تُشير إلى أن محرر Apps Script يمكن أن يحصل في النهاية على مساعدة مدمجة في توليد الكود وتصحيح الأخطاء ومراجعته. هذا من شأنه تبسيط سير العمل بشكل كبير، مما يجعل النهج الحالي القائم على واجهة برمجة التطبيقات مقدمة لتجربة أكثر سلاسة.**

**مع تزايد انتشار الذكاء الاصطناعي في توليد الكود، تُصبح الاعتبارات الأخلاقية (المساءلة، التحيز، الملكية الفكرية، الثغرات الأمنية) حاسمة بشكل متزايد. لن يتعلق مستقبل تطوير Apps Script باستخدام الذكاء الاصطناعي بما يمكن أن يفعله الذكاء الاصطناعي فحسب، بل بمدى مسؤوليته في استخدامه. هذا يعني أن مبادئ وممارسات "الذكاء الاصطناعي المسؤول" ستصبح كفاءة أساسية لمطوري Apps Script، متجاوزة مجرد الكفاءة التقنية لتشمل الاعتبارات الأخلاقية في دورة حياة التطوير الخاصة بهم.**

**المراجع المستخدَمة في التقرير**