

11 · البرمجة والمنصات

قائمة مراجعة الكود الناتج بالذكاء الاصطناعي

ما تفحصه في كود كتبه الذكاء الاصطناعي قبل دمج: المدخلات، الصلاحيات، الأسرار، الاعتماديات، الأخطاء، الاختبارات، والنشر، مع مثالين مشروحين.

لمن: لمن يبني مواقع أو أدوات بمساعدة الذكاء الاصطناعي ويريد أن يعرف ما يجب فحصه قبل أن يصل الكود إلى مستخدمين حقيقيين، سواء كان مطورًا مبتدئًا أو صاحب مشروع يراجع مع فريقه.

المستوى: متوسط الإصدار 1.0 · 21-09-2026

الكود الذي يكتبه الذكاء الاصطناعي يعمل غالبًا في «المسار السعيد»: المستخدم يعي الحقول كما يجب، ويطلب بياناته هو، والشبكة لا تنقطع. المشاكل تظهر في كل ما عدا ذلك: مستخدم يغيّر رقمًا في الرابط، حقل فارغ، رسالة خطأ تكشف تفاصيل الخادم، مفتاح سرّي منسوخ في الملف. هذه القائمة تمنحك أسئلة ثابتة تطرحها على كل تغيير قبل أن يصل إلى مستخدمين حقيقيين.

تنبيه مهم: هذه القائمة ليست بديلًا عن تدقيق أمني احترافي. هي حد أدنى يلتقط الأخطاء الشائعة. أي منصة تحمل بيانات شخصية حساسة (صحية، مالية، عن قاصرين) أو تستقبل مدفوعات تحتاج مراجعة من مختص أمن قبل الإطلاق وبعد التغييرات الكبيرة.

طريقة المراجعة

- اقرأ ما تغيّر، لا المشروع كله. راجع الفرق بين النسخة السابقة والجديدة ملفًا بملفًا. التغيير الكبير جدًا يُقسّم قبل مراجعته.
- اطلب من الأداة أن تشرح. «اشرح ما يفعله كل جزء، وما الافتراضات التي بنيت عليها، وما المدخلات التي قد تكسره». إن لم تفهم الشرح، لا تدمج الكود.
- شغل الكود بنفسك. المسار السعيد أولًا، ثم المسارات غير السعيدة: حقل فارغ، قيمة طويلة جدًا، مستخدم غير مسجّل، مستخدم آخر.
- مرّ على الأقسام التسعة أدناه. ضع ✓ أو ✗ أو «لا ينطبق» لكل بند.
- سجّل القرار: يُدمج، يُعدّل، أو يُرفض، مع ما تعلّمته لتضيفه إلى تعليماتك الدائمة للأداة.

1. الوظيفة

- ☐ الكود يفعل ما طُلب فقط، ولم يضيف ميزات أو ملفات أو صفحات لم تُطلب.
- ☐ جربته فعليًا على جهازك أو بيئة الاختبار، ولم تكتفِ بقول الأداة إنه «يعمل».
- ☐ الحالات الحدية مغطاة: قائمة فارغة، عنصر واحد، قيم كبيرة جدًا، تواريخ في مناطق زمنية مختلفة.
- ☐ لم يحذف الكود أو يعطل شيئًا كان يعمل (تحقق من الملفات التي عُدلت ولم تتوقع تعديلها).

☐ لا دوال أو مكتبات «مختلقة» غير موجودة فعلاً.

2. التحقق من المدخلات

- ☐ كل ما يأتي من المستخدم (نماذج، روابط، ملفات، رؤوس الطلب) يُعامل كغير موثوق.
- ☐ التحقق يتم في الخادم، لا في المتصفح فقط؛ تحقق المتصفح راحة للمستخدم لا حماية.
- ☐ لكل حقل: النوع، الطول الأقصى، الصيغة المسموحة، وهل هو إلزامي.
- ☐ الاستعلامات إلى قاعدة البيانات مُعلّمة المعاملات (parameterized)، لا نصوص مركّبة من مدخلات المستخدم.
- ☐ ما يُعرض من مدخلات المستخدم في الصفحة يُهرَّب (escaped) ولا يُدرج كـ HTML خام.
- ☐ رفع الملفات محدود بالنوع والحجم، ولا يُنقذ أي ملف مرفوع.

3. تسجيل الدخول والصلاحيات

- ☐ كل مسار يعرض أو يعدّل بيانات يتحقّق أولاً أن المستخدم مسجّل.
- ☐ ثم يتحقّق أن هذا المستخدم يملك هذا السجل تحديداً أو له الدور المطلوب (راجع مصفوفة الصلاحيات إن وُجدت).
- ☐ جرّبت تغيير المعرّف في الرابط أو الطلب إلى معرّف مستخدم آخر، والنتيجة رفض.
- ☐ إخفاء زر في الواجهة ليس صلاحية؛ الخادم يرفض الطلب ولو أُرسل مباشرة.
- ☐ في قواعد البيانات التي تدعم سياسات أمان على مستوى الصف، السياسات مفعّلة ومختبرة بمستخدمين مختلفين.
- ☐ الردود لا تُرجع حقولاً لا يحتاجها المستخدم (كلمات مرور مشقّرة، ملاحظات داخلية، بيانات مستخدمين آخرين).

4. الأسرار

- ☐ لا مفاتيح أو كلمات مرور أو رموز وصول مكتوبة في الكود؛ كلها في متغيّرات بيئة أو مخزن أسرار.
- ☐ ملف متغيّرات البيئة المحلي مستثنى من المستودع، وهناك ملف مثال بأسماء المتغيّرات فقط.
- ☐ لا أسرار في كود المتصفح: كل ما يُرسل إلى المتصفح يراه أي زائر.
- ☐ المفاتيح المستخدمة بصلاحيات محدودة قدر الإمكان (مفتاح للقراءة فقط إن كفى).
- ☐ إن ظهر مفتاح في الكود أو في محادثة مع أداة أو في سجل، فقد انكشف: غيّره فوراً، ولا يكفي حذفه.

5. الاعتماديات

- ☐ كل مكتبة جديدة أضافتها الأداة موجودة فعلاً، باسمها الصحيح حرفياً، ومن مصدرها الرسمي (الأسماء المتشابهة فخّ معروف).
- ☐ المكتبة ضرورية فعلاً، ولا يكفي كود قصير أو مكتبة موجودة مسبقاً في المشروع.
- ☐ المكتبة تُصان ولها تحديثات حديثة، وأداة فحص الثغرات في مدير الحزم لا تُظهر مشاكل خطيرة.
- ☐ ملف قفل الإصدارات محدّث ومرفوع مع التغيير.

6. الأخطاء والسجلات

- ☐ الأخطاء تُلتقط ولا تُسقط التطبيق كله.
- ☐ رسالة الخطأ للمستخدم واضحة وبلغته، ولا تكشف تفاصيل تقنية (مسارات ملفات، استعلامات، إصدارات).
- ☐ السجلات تحتوي ما يكفي للتشخيص، ولا تحتوي كلمات مرور أو مفاتيح أو بيانات شخصية كاملة.
- ☐ العمليات الخارجية (بريد، دفع، واجهات برمجية) لها مهلة وسلوك واضح عند الفشل.
- ☐ لا أخطاء «تُبتلع» بصمت بكتلة التقاط فارغة.

7. الاختبارات

- ☐ للتغيير اختبار واحد على الأقل للمسار السعيد وواحد لمسار الرفض (مستخدم غير مخوّل، مدخل غير صالح).
- ☐ الاختبارات تفشل فعلاً حين تكسر الكود عمداً (اختبار لا يفشل أبداً لا يختبر شيئاً).
- ☐ الأداة لم «تصلح» اختباراً فاشلاً بتعديل الاختبار ليتوافق مع الخطأ.
- ☐ كل الاختبارات السابقة ما زالت تنجح.

8. إتاحة الوصول (للوجهات)

- ☐ عناصر تفاعلية حقيقية (زر للأفعال، رابط للتنقّل) لا عناصر عامة بنقرة مضافة.
- ☐ كل حقل له عنوان مرتبط به، ورسائل الخطأ مقروءة لقارئ الشاشة.
- ☐ يمكن الوصول إلى كل شيء بلوحة المفاتيح، ومؤشّر التركيز ظاهر.
- ☐ الصور لها نص بديل، والتباين كافٍ.
- ☐ الاتجاه من اليمين إلى اليسار صحيح في الصفحات العربية، واللغة معلنة في الصفحة.

9. النشر

- ☐ متغيّرات البيئة مضبوطة في بيئة الإنتاج، وليست قيم الاختبار.
- ☐ وضع التصحيح (debug) معطل في الإنتاج.
- ☐ تغييرات قاعدة البيانات (migrations) مراجعة، ولا تحذف بيانات، وهناك نسخة احتياطية حديثة قبل تطبيقها.
- ☐ هناك طريقة معروفة للتراجع إلى النسخة السابقة إن فشل النشر.
- ☐ بعد النشر: فحص سريع للمسارات الأساسية على الموقع الحقيقي.

مثال مشروح 1: مسار يعرض حجزاً

مثال توضيحي

تطبيق وهمي لـ «عيادة نبض» يعرض للمريض تفاصيل حجزه. طُلب من أداة ذكاء اصطناعي «مسار يعيد بيانات الحجز حسب رقمه»، فكتبت الكود التالي (بأسلوب Express، مبسّط للتوضيح):

```
GET /api/bookings/:id //
} <= app.get('/api/bookings/:id', async (req, res)
const booking = await db.bookings.findById(req.params.id); // [1]
res.json(booking); // [2]
;({

const SMS_API_KEY = 'sk_live_EXAMPLE_ONLY'; // [3]
```

- **[1] لا تحقق من الهوية ولا من الملكية.** أي شخص، مسجلًا أو لا، يستطيع تجربة أرقام متتالية وقراءة حجوزات كل المرضى. هذا من أكثر الأخطاء شيوعًا وخطورة.
- **[2] يُرجع السجل كاملاً.** قد يتضمن رقم الهاتف والملاحظات الداخلية للمعالج. وإن لم يوجد الحجز يُرجع قيمة فارغة بلا رسالة واضحة.
- **[3] مفتاح سرّي داخل الكود.** سيدخل المستودع وكل نسخة منه. (المفتاح هنا وهمي للتوضيح).

```
GET /api/bookings/:id //
app.get('/api/bookings/:id', requireLogin, async (req, res) => { // [1]
;const booking = await db.bookings.findById(req.params.id)
if (!booking || booking.patientId !== req.user.id) { // [2]
;return res.status(404).json({ error: 'Not found' })
{
res.json({ // [3]
, id: booking.id
, date: booking.date
, status: booking.status
;({
;({

const SMS_API_KEY = process.env.SMS_API_KEY; // [4]
```

- **[1] المسار لا يعمل إلا لمستخدم مسجل.**
 - **[2] الحجز يجب أن يخصّ هذا المستخدم.** الرد 404 نفسه في الحالتين، فلا يعرف المهاجم إن كان الرقم موجودًا.
 - **[3] تُرجع الحقول التي تحتاجها الواجهة فقط.**
 - **[4] المفتاح في متغيّر بيئة.** وإن كان المفتاح الحقيقي قد رُفِع سابقًا إلى المستودع، فيجب تغييره لدى المزود لا حذفه من الكود فقط.
- كيف نختبر:** سجّل الدخول بمستخدمين، واطلب بالمستخدم الأول رقم حجز يخصّ الثاني؛ النتيجة المتوقعة 404. ثم اطلب بلا تسجيل دخول؛ النتيجة رفض.

مثال مشروح 2: نموذج تواصل

مثال توضيحي

موقع وهمي لـ «مخبز الزيتون» فيه نموذج تواصل. الكود الذي أنتجته الأداة:

قبل المراجعة

```
POST /api/contact //
} <= app.post('/api/contact', async (req, res)
const { name, phone, message } = req.body; // [1]
)await db.query
+ ") INSERT INTO messages (name, phone, message) VALUES"
name + ", " + phone + ", " + message + ")" // [2]
;(
res.send('Thanks ' + name); // [3]
;{
```

- **[1] لا تحقق.** حقول فارغة، أو رسالة بطول مليون حرف، أو رقم هاتف عشوائي، كلها مقبولة. ولا حدّ لعدد الإرسالات، فيسهل إغراق الجدول برسائل مزعجة.
- **[2] استعلام مركّب من مدخلات المستخدم.** هذا باب «حقن SQL»: مدخل مصمّم بعناية قد يقرأ أو يحذف بيانات.
- **[3] يعرض الاسم كما هو داخل رد HTML.** اسم يحتوي شيفرة قد يُنفَّذ في المتصفح.

بعد المراجعة

```
POST /api/contact (rateLimit: illustrative, use your framework's limiter) //
} <= app.post('/api/contact', rateLimit({ max: 5, perMinutes: 10 }), async (req, res)
const name = String(req.body.name || "").trim(); // [1]
);const phone = String(req.body.phone || "").trim
);const message = String(req.body.message || "").trim
);const phoneOk = /^[0-9+ ]{7,20}$/.test(phone)
} if (!name || name.length > 80 || !phoneOk || message.length > 1000)
;return res.status(400).json({ error: 'invalid_input' })
{
await db.query( // [2]
,'INSERT INTO messages (name, phone, message) VALUES ($1, $2, $3)'
[name, phone, message]
);(
res.json({ ok: true }); // [3]
;{
```

- **[1] كل حقل يُحوّل إلى نص ويُقَصّ ويُفحص طوله وصيغته في الخادم، مع حدّ لعدد الإرسالات من المصدر نفسه.**
- **[2] استعلام بمعاملات:** القيم تُمرّر منفصلة عن نص الاستعلام، فلا تُفسّر كأوامر.
- **[3] الرد بيانات لا HTML، والواجهة تعرض رسالة الشكر بنصّها الخاص، ولا تُدرج مدخلات المستخدم كـ HTML.**

كيف نختبر: أرسل النموذج فارغاً، وبرسالة طويلة جداً، وبرقم هاتف فيه حروف؛ النتيجة المتوقعة رفض برسالة واضحة في الواجهة. أرسل ست مرات متتالية؛ السادسة تُرفض.

برومبت المراجعة الذاتية

استخدمه بعد أن تكتب الأداة الكود، في محادثة جديدة أو مع أداة ثانية. النتيجة مساعدة لك لا حكم نهائي؛ القرار لمن يراجع.

برومبت مراجعة الكود

راجع الكود التالي كمراجع أمان صارم. لا تعد كتابته كاملاً.
(1) اشرح ما يفعله في 5 أسطر.
(2) لكل مسار: هل يتحقق من تسجيل الدخول؟ من الملكية أو الدور؟
(3) أين تُستخدم مدخلات المستخدم دون تحقق أو دون تهريب؟
(4) هل توجد أسرار أو إعدادات مكتوبة في الكود؟
(5) ماذا يحدث عند فشل كل عملية خارجية؟
(6) ما الاختبارات الثلاثة الأهم التي تنقص؟
رتب الملاحظات من الأخطر إلى الأقل، واذكر رقم السطر.
الكود: [الصق الكود بعد استبدال أي مفتاح أو بيانات حقيقية بقيم وهمية]

متى تتوقف وتطلب مختصاً

- المنصة تخزن بيانات صحية أو مالية أو بيانات قاصرين.
- تستقبل مدفوعات أو تدير أرصدة.
- بنيت نظام تسجيل دخول أو تشفير بنفسك بدل استخدام خدمة أو مكتبة معروفة.
- لم تستطع فهم شرح الأداة لجزء حساس من الكود.
- ظهر سلوك غريب لا تفسير له في السجلات أو في قاعدة البيانات.

أخطاء شائعة

- **الخطأ:** دمج الكود لأنه «يعمل» في التجربة الأولى. **الحل:** «يعمل» تعني المسار السعيد فقط؛ اختبر الرفض والحالات الحدية.
- **الخطأ:** الاعتماد على إخفاء الأضرار كصلاحيات. **الحل:** الخادم يرفض كل طلب غير مخوّل ولو أُرسل مباشرة.
- **الخطأ:** لصق مفاتيح حقيقية في المحادثة مع الأداة «لتعمل بسرعة». **الحل:** استخدم أسماء متغيّرات وقيماً وهمية، وإن حدث التسريب غير المفتاح.
- **الخطأ:** قبول مكتبة جديدة لم تسمع بها. **الحل:** تحقق من الاسم الحرفي والمصدر والصيانة قبل التثبيت.
- **الخطأ:** مراجعة تغيير ضخّم دفعة واحدة. **الحل:** اطلب من الأداة تغييرات صغيرة، وراجع كل واحد على حدة.
- **الخطأ:** ترك الأداة تعدّل الاختبار ليتوافق مع الخطأ. **الحل:** الاختبار يمثل المطلوب؛ الكود هو ما يُصلح.

قائمة الإنجاز قبل الدمج

- ☐ فهمت ما يفعله الكود وجربته بنفسك.
- ☐ المدخلات مفحوصة في الخادم، والاستعلامات بمعاملات.
- ☐ كل مسار يتحقق من الهوية ومن الملكية أو الدور، وجربت الوصول بمستخدم آخر.
- ☐ لا أسرار في الكود أو في كود المتصفح أو في السجلات.
- ☐ الاعتماديات الجديدة حقيقية وضرورية ومصانة.
- ☐ رسائل الخطأ لا تكشف تفاصيل تقنية.
- ☐ اختبار نجاح واختبار رفض على الأقل، وكل الاختبارات السابقة تنجح.
- ☐ الواجهات قابلة للاستخدام بلوحة المفاتيح وبقارئ الشاشة.
- ☐ النشر له نسخة احتياطية وطريقة تراجع.
- ☐ إن كانت البيانات حساسة: موعد مراجعة أمنية احترافية محدد.

عن المُعَدِّ — محمد شرف الدين، رائد أعمال من طرابلس، لبنان. مؤسس وكالة باشن للتسويق الرقمي، ورئيس جمعية تكافل الشبابية، ومؤسس منصات رقمية. يدرب الأفراد والفرق على الذكاء الاصطناعي التطبيقي والتسويق.

الأمثلة في هذا الملف خيالية للتوضيح. يمكنك استخدامه في عملك ومؤسستك مجاناً مع ذكر المصدر عند إعادة النشر.

[/mohamadcharafeddine.com/ar/resources/ai-code-review-checklist](https://mohamadcharafeddine.com/ar/resources/ai-code-review-checklist)