# 🧪 أسئلة متقدمة في Software Testing مع الإجابات

## 🧩 1. ما الفرق بين اختبار الدخان (Smoke Testing) واختبار الصحة (Sanity Testing)؟

**الجواب:**

- **Smoke Testing**: اختبار أساسي سريع يتم بعد كل Build للتأكد من أن التطبيق مستقر بدرجة كافية للاختبار الأعمق.
- **Sanity Testing**: يتم بعد إصلاح عطل أو إضافة ميزة للتأكد من أن التغييرات تعمل بشكل صحيح دون التحقق من بقية النظام.

---

## 🧩 2. كيف تختلف اختبارات Unit عن اختبارات Integration؟

**الجواب:**

- **Unit Testing**: اختبار وظائف أو وحدات صغيرة من الكود بشكل مستقل.
- **Integration Testing**: التأكد من أن وحدات النظام تعمل معًا بشكل صحيح، وتُركز على واجهات التفاعل بين المكونات.

---

## 🧩 3. ما هو الـ Test Pyramid؟ ولماذا هو مهم؟

**الجواب:**
هرم الاختبار يشير إلى توزيع الاختبارات:

- القاعدة: **Unit Tests** (أكثر عددًا، أسرع).
- الوسط: **Service/Integration Tests**.
- القمة: **UI Tests** (أقل عددًا، أبطأ).
  مهم لأنه يشجع على كتابة اختبارات قابلة للصيانة وفعالة من حيث الأداء.

---

### 🧩 4. كيف تُنفّذ اختبارات الـ API؟

**الجواب:**

1. استخدام أدوات مثل Postman أو RestAssured.
2. تحديد الـ endpoints، طرق الطلب (GET, POST)، البيانات، الهيدر.
3. التحقق من الاستجابة، الكود (200/400/500)، والبيانات المتوقعة.

---

### 🧩 5. ما هو الفرق بين TDD و BDD؟

**الجواب:**

- **TDD (Test-Driven Development)**: تكتب الاختبار قبل الكود.
- **BDD (Behavior-Driven Development)**: تركز على السلوك باستخدام لغة قريبة من الأعمال (مثل Gherkin).
  BDD أسهل للفريق لفهمه ويشجع التعاون بين QA و Dev و Business.

---

### 🧩 6. ما هو Flaky Test؟ وكيف تتعامل معه؟

**الجواب:**
اختبار يعطي نتائج غير مستقرة (ينجح ويفشل عشوائيًا).
الحل: تحليل السبب (مثل مشاكل الـ async، الاعتماد على حالة البيئة)، وإعادة كتابته بشكل أكثر موثوقية.

---

### 🧩 7. كيف تتحقق من أداء النظام؟

**الجواب:**
من خلال **Performance Testing** باستخدام أدوات مثل JMeter أو Gatling لقياس:

- الاستجابة (Response Time).
- التحميل (Throughput).
- السعة القصوى (Scalability).

---

### 🧩 8. ما الفرق بين Functional و Non-Functional Testing؟

**الجواب:**

- **Functional Testing**: يتحقق من الوظائف حسب المتطلبات.
- **Non-Functional Testing**: يتحقق من الأداء، الأمان، قابلية الاستخدام، إلخ.

---

### 🧩 9. ما أهمية Regression Testing؟

**الجواب:**
يضمن أن التغييرات الجديدة لم تكسر الوظائف القديمة. يُنفّذ بعد كل تعديل أو تحديث لضمان استقرار النظام.

---

### 🧩 10. ما هي أنواع الـ Test Coverage؟

**الجواب:**

1. **Statement coverage**
2. **Branch coverage**
3. **Path coverage**
   تُستخدم لقياس مدى اختبار الكود، وتعزيز جودة الاختبارات.

### 🧩 11. ما هو Test Oracle؟

**الجواب:**
هو آلية أو مصدر يُستخدم لمعرفة إذا كانت نتائج اختبار ما صحيحة أم لا. يمكن أن يكون وثائق، مواصفات النظام، أو نتائج من أنظمة مشابهة.

---

### 🧩 12. ما هو مفهوم Mocking ولماذا نستخدمه؟

**الجواب:**
Mocking هو إنشاء نسخة مزيفة من كائن أو خدمة لاختبار وحدة معينة دون الاعتماد على الطرف الحقيقي. مهم لاختبار وحدات معزولة والتحكم في السيناريوهات.

---

### 🧩 13. ما الفرق بين Verification و Validation؟

**الجواب:**

- **Verification**: هل بنينا النظام بشكل صحيح؟ (التحقق من التصميم/المواصفات).
- **Validation**: هل بنينا النظام الصحيح؟ (اختبار المنتج نفسه لتلبية الاحتياجات).

---

### 🧩 14. كيف تتعامل مع اختبارات تعتمد على الوقت (Time-sensitive tests)؟

**الجواب:**
استخدام **time mocking** أو **freezing time** (مثل مكتبة `freezegun` في بايثون)، أو الاعتماد على مهام cron وهمية لتثبيت الزمن.

---

### 🧩 15. ما هي Traceability Matrix؟

**الجواب:**
جدول يربط بين متطلبات النظام وحالات الاختبار المرتبطة بها. يساعد في التأكد من أن كل متطلب تم اختباره.

---

### 🧩 16. اشرح مفهوم Shift-Left Testing.

**الجواب:**
نقل أنشطة الاختبار مبكرًا في دورة التطوير (خلال مراحل التحليل أو التصميم)، مما يساعد على اكتشاف العيوب مبكرًا وتقليل تكلفة الإصلاح.

---

### 🧩 17. ما الفرق بين Stub و Mock؟

**الجواب:**

- **Stub**: يُرجع قيم ثابتة لتكملة الاختبار.
- **Mock**: يتحقق من التفاعل مع الكائن (الاستدعاءات والقيم).

---

### 🧩 18. كيف تختبر نظام موزّع Distributed System؟

**الجواب:**

- اختبار تزامن البيانات والـ consistency.
- اختبار failure scenarios (شبكة مقطوعة، service down).
- اختبار latency وreplication delays.

---

## 🧩 19. ما هو الفرق بين Load Testing و Stress Testing؟

**الجواب:**

- **Load Testing**: اختبار النظام تحت حمل طبيعي أو متزايد تدريجيًا.
- **Stress Testing**: دفع النظام لأقصى حدوده لكشف نقاط الضعف.

---

## 🧩 20. متى تستخدم Exploratory Testing؟

**الجواب:**
عندما لا تكون هناك حالات اختبار محددة مسبقًا، أو لاكتشاف أخطاء غير متوقعة. يُستخدم في المراحل المتأخرة أو مع نظم جديدة.

---

## 🧩 21. ما هو الـ Test Plan وما مكوناته؟

**الجواب:**
وثيقة تحدد نطاق، نهج، موارد، وجدول اختبار المشروع. تشمل:

- أهداف الاختبار.
- ما سيتم اختباره / استبعاده.
- بيئة الاختبار.
- المسؤوليات والمخاطر.

---

## 🧩 22. ما هي تقنية Equivalence Partitioning؟

**الجواب:**
تقسيم المدخلات إلى مجموعات بحيث يتم اختبار حالة واحدة من كل مجموعة، لأنها تمثل الآخرين. تقلل عدد الحالات وتزيد الكفاءة.

---

### 🧩 23. ما هي Boundary Value Analysis؟

**الجواب:**
اختبار القيم الحدّية (أعلى وأدنى) لأن معظم الأخطاء تحدث عند الحدود. مثال: إذا المدى 1–100، نختبر 0,1,100,101.

---

### 🧩 24. ما هو مفهوم Test Environment؟

**الجواب:**
هو البيئة التقنية التي تُنفّذ فيها الاختبارات، وتشمل الأجهزة، الأنظمة، قواعد البيانات، الإعدادات... يجب أن تماثل بيئة الإنتاج قدر الإمكان.

---

### 🧩 25. ما الفرق بين Defect, Bug, Error, Fault؟

**الجواب:**

- **Error**: خطأ بشري في الكود.
- **Fault**: نتيجة هذا الخطأ في البرنامج.
- **Bug**: مصطلح عام يُستخدم لاكتشاف خلل.
- **Defect**: فرق بين النتيجة المتوقعة والفعلية.

---

### 🧩 26. ما هو Retesting؟

**الجواب:**
إعادة تنفيذ نفس الحالة بعد إصلاح خلل للتأكد من أن العطل قد تم إصلاحه فعلًا.

---

### 🧩 27. كيف تُقيّم جودة الاختبارات Automated؟

**الجواب:**
من خلال:

- نسبة التغطية (Coverage).
- موثوقية النتائج.

- سرعة التنفيذ.
- سهولة التعديل والصيانة.

---

### 🧩 28. كيف تتعامل مع اختبارات تعتمد على بيانات متغيرة (Dynamic Data)؟

**الجواب:**
استخدام **test data generators**، أو بيانات عشوائية، أو إعادة ضبط البيانات قبل كل اختبار لضمان الاستقرار.

---

### 🧩 29. ما هي الـ Heuristic Testing؟

**الجواب:**
اختبار يعتمد على خبرة المختبر واستخدام قواعد عامة أو "حيل ذكية" للكشف عن الأعطال دون حالات اختبار محددة.

---

### 🧩 30. كيف تصمم Test Cases جيدة؟

**الجواب:**

- تغطي جميع المسارات المحتملة.
- سهلة الفهم والتنفيذ.
- قابلة لإعادة الاستخدام.
- تحتوي على مدخلات/مخرجات واضحة.
- تسجل خطوات إعادة الاختبار.

### 🧩 31. ما الفرق بين Test Case و Test Scenario؟

**الجواب:**

- **Test Case**: يحتوي على خطوات تفصيلية، بيانات إدخال، ونواتج متوقعة.
- **Test Scenario**: تمثيل عالي المستوى لحالة أو وظيفة لاختبارها (مثل: "تحقق من عملية تسجيل الدخول").

---

### 🧩 32. ما هي مبادئ الـ CI/CD Testing؟

**الجواب:**

- تنفيذ اختبارات تلقائية بعد كل commit أو deployment.
- تسهيل الاكتشاف المبكر للأخطاء.
- دمج اختبارات Unit، Integration، و End-to-End في خطوط الـ pipeline.

---

## 🧩 33. اشرح ما هو Mutation Testing.

**الجواب:**
هو إدخال تغييرات طفيفة في الكود (mutants) والتحقق من أن الاختبارات تلتقط هذه التغييرات. إذا لم تفشل الاختبارات، فهي ضعيفة.

---

## 🧩 34. ما أهمية Negative Testing؟

**الجواب:**
تضمن أن النظام يتعامل بشكل صحيح مع مدخلات غير صالحة أو غير متوقعة، وتمنع الأعطال والانهيارات.

---

## 🧩 35. كيف تختبر REST API من منظور الأمن؟

**الجواب:**

- تحقق من التفويض والمصادقة (Auth).
- اختبر rate limiting.
- تحقق من منع إدخال بيانات ضارة (SQL/XSS).
- تحقق من headers مثل CORS, Content-Type.

---

## 🧩 36. ما الفرق بين Test Suite و Test Plan؟

**الجواب:**

- **Test Plan**: وثيقة تخطيطية شاملة للاختبار.
- **Test Suite**: مجموعة من الحالات المجدولة للتنفيذ.

---

### 🧩 37. ما هو A/B Testing؟

**الجواب:**
تقنية لاختبار نسختين من منتج (A و B) على مستخدمين مختلفين لمعرفة أي نسخة تحقق أداءً أفضل (يُستخدم بكثرة في UX).

---

### 🧩 38. ما المقصود بـ Code Coverage؟ وهل تعني اختبارات فعّالة؟

**الجواب:**
هي نسبة الكود الذي تم اختباره فعليًا. لكنها لا تضمن الجودة دائمًا، فربما نغطي الكود لكن دون التحقق من النتائج المتوقعة بدقة.

---

### 🧩 39. ما هو Test Debt؟

**الجواب:**
اختبارات غير مكتملة أو مفقودة تُؤجّل لاحقًا لتسريع التطوير، لكنها تسبب مشاكل مستقبلية.

---

### 🧩 40. كيف تُحدد أولويات الاختبارات؟

**الجواب:**

- حسب المخاطر (Risk-based Testing).
- أهمية الوظيفة.
- التكرار في الاستخدام.
- الأعطال السابقة.

---

### 🧩 41. ما هو مفهوم End-to-End Testing؟

**الجواب:**
اختبار تدفق كامل من البداية للنهاية كما يفعل المستخدم، للتحقق من أن النظام يتكامل ويعمل كوحدة واحدة.

---

### 🧩 42. كيف تختبر نظام يعتمد على الطرف الثالث (3rd Party API)؟

**الجواب:**

- استخدام stubs/mocks.

- التأكد من التعامل مع الـ timeouts والأخطاء.
- مراجعة القيود والسياسات (مثل quotas).

---

### 🧩 43. ما هو Static Testing؟ وهل له قيمة؟

**الجواب:**
اختبار بدون تنفيذ الكود، مثل مراجعة الكود أو المواصفات. يساعد في اكتشاف الأخطاء مبكرًا ويوفر وقت وجهد كبيرين.

---

### 🧩 44. ما هي قابلية الاختبار (Testability)؟

**الجواب:**
هي مدى سهولة اختبار النظام، وتشمل وضوح الكود، سهولة توليد البيانات، إمكانية مراقبة النتائج، وتقارير الأخطاء المفهومة.

---

### 🧩 45. ما هي العوامل التي تؤثر على قرار أتمتة اختبار معين؟

**الجواب:**

- تكرار الاختبار.
- مدى استقراره.
- الوقت والتكلفة.
- تعقيده ومدى فائدته.

---

### 🧩 46. ما هو الفرق بين Bug Severity و Priority؟

**الجواب:**

- **Severity**: مدى تأثير الخطأ (عالية = النظام لا يعمل).
- **Priority**: مدى إلحاح إصلاحه (عالية = لازم يتصلح فورًا حتى لو تأثيره بسيط).

---

### 🧩 47. ما هي اختبارات Accessibility؟

**الجواب:**
تختبر إذا كان التطبيق قابلًا للاستخدام من قبل ذوي الاحتياجات الخاصة (مثل قارئات الشاشة، التنقل عبر لوحة المفاتيح، تباين الألوان).

---

### 🧩 48. ما هو الفرق بين QA و QC؟

**الجواب:**

- **QA (Quality Assurance)**: الوقاية من الأخطاء (تحسين العمليات).
- **QC (Quality Control)**: اكتشاف الأخطاء (عبر الاختبار والتحقق).

---

### 🧩 49. ما هي أهمية Continuous Testing؟

**الجواب:**
دمج الاختبار في كل مراحل دورة الحياة لتوفير تغذية راجعة سريعة، تحسين الجودة، وتقليل الوقت للوصول للسوق.

---

### 🧩 50. كيف تقيس فعالية فريق الاختبار؟

**الجواب:**

- نسبة العيوب المكتشفة قبل الإنتاج.
- سرعة الاستجابة لتغييرات المتطلبات.
- تغطية المتطلبات.
- نسبة العيوب المتكررة.

### 🧩 51. ما هو Risk-Based Testing؟

**الجواب:**
هو أسلوب يركز على اختبار الأجزاء الأكثر عرضة للفشل أو التي تحمل أكبر تأثير على العمل، بهدف استخدام الموارد بكفاءة وتقليل المخاطر.

---

### 🧩 52. ما الفرق بين Alpha و Beta Testing؟

**الجواب:**

- **Alpha**: يُجرى داخليًا بواسطة فريق الشركة قبل الإطلاق.
- **Beta**: يُجرى خارجيًا بواسطة مستخدمين حقيقيين لتجربة المنتج في ظروف واقعية.

---

### 🧩 53. متى تستخدم Data-Driven Testing؟

**الجواب:**
عند الحاجة لاختبار نفس السيناريو بمدخلات متعددة. تُستخدم ملفات Excel/CSV/JSON كمصادر بيانات، وتُفيد في تقليل التكرار.

---

### 🧩 54. ما هو الفرق بين Assertion و Verification؟

**الجواب:**

- **Assertion**: تأكيد برمجي يُستخدم داخل كود الاختبار.
- **Verification**: خطوة تحقق من التوقعات (يدوية أو تلقائية).

---

### 🧩 55. ما هي تقنية Pairwise Testing؟

**الجواب:**
اختبار التركيبات الثنائية من المدخلات (wise-2) لتقليل عدد الحالات مع الحفاظ على تغطية فعالة لمعظم الأخطاء المحتملة.

---

### 🧩 56. ما المقصود بـ Defect Clustering؟

**الجواب:**
فكرة أن معظم الأخطاء تتجمع في عدد قليل من الوحدات، ويُستند إليها في استهداف الاختبارات بناءً على خبرة أو بيانات سابقة.

---

### 🧩 57. ما الفرق بين Use Case Testing و Scenario Testing؟

**الجواب:**

- **Use Case**: اختبارات مبنية على حالات استخدام موثقة.
- **Scenario Testing**: سيناريوهات كاملة تغطي تفاعلات متعددة تشمل وظائف متعددة.

---

### 🧩 58. ما أهمية Logging في الاختبارات؟

**الجواب:**
يساعد في تتبع الفشل، فهم السبب الجذري، وإعادة بناء الحالة. مهم جدًا في الاختبارات التلقائية والبيئات المعقدة.

---

### 🧩 59. كيف تختبر تطبيقات Microservices؟

**الجواب:**

- اختبار كل خدمة بشكل مستقل (contract testing).
- اختبار الاتصالات بين الخدمات (integration testing).
- محاكاة الفشل الجزئي (chaos testing).

---

### 🧩 60. ما هي Fault Injection Testing؟

**الجواب:**
تقنية يتم فيها إدخال أخطاء صناعية (مثل قطع الاتصال أو ارتفاع الذاكرة) لاختبار مدى استجابة النظام وتحمله.

---

### 🧩 61. ما هو الفرق بين Usability و User Experience Testing؟

**الجواب:**

- **Usability**: قابلية الاستخدام (سهولة التنقل، وضوح النصوص).
- **UX**: يشمل المشاعر والانطباع العام عن التفاعل مع النظام.

---

### 🧩 62. ما هي عيوب الاعتماد الكلي على الاختبارات التلقائية؟

**الجواب:**

- قد تفشل في اكتشاف مشاكل تجربة المستخدم.
- صعبة الصيانة إذا تغير UI.

- لا تغني عن التفكير الإبداعي والاختبارات الاستكشافية.

---

### 🧩 63. كيف تختبر تقارير تحتوي على رسوم بيانية؟

**الجواب:**

- التحقق من الدقة الحسابية.
- توافق القيم مع البيانات الخام.
- عرض صحيح للرسومات (محاور، ألوان، تفاعل المستخدم).

---

### 🧩 64. ما الفرق بين Manual و Exploratory Testing؟

**الجواب:**

- **Manual**: اختبارات مخططة مسبقًا تنفذ يدويًا.
- **Exploratory**: اختبارات غير مخططة، تعتمد على الخبرة والاستكشاف أثناء الاستخدام.

---

### 🧩 65. ما أهمية تحليل العطل Root Cause Analysis؟

**الجواب:**
يساعد في منع تكرار الأخطاء، تحسين جودة الكود والاختبارات، وتقليل الوقت الضائع في تصحيح نفس النوع من المشاكل.

---

### 🧩 66. ما هو الـ AQL في فحص الجودة؟

**الجواب:**
**Acceptable Quality Level**: هو الحد الأقصى المقبول للأخطاء ضمن عينة، ويُستخدم في اختبار الدُفعات وفقًا للمعايير الصناعية.

---

### 🧩 67. كيف تُقيّم أدوات الاختبار التلقائي؟

**الجواب:**
حسب:

- دعم اللغات/المنصات.
- سهولة الاستخدام والتعلم.
- سرعة التنفيذ.
- التكامل مع CI/CD.
- دعم المجتمع والمصادر.

---

### 🧩 68. كيف تتعامل مع flaky UI tests؟

**الجواب:**

- استخدام `waitFor` بدلاً من التأخيرات الثابتة.
- استخدام معرفات مستقرة (IDs).
- تقليل الاعتماد على التوقيت أو الأنيميشن.

---

### 🧩 69. ما الفرق بين Smoke و Regression Testing؟

**الجواب:**

- **Smoke**: فحص سريع لأهم الوظائف.
- **Regression**: فحص شامل لضمان أن كل شيء لا يزال يعمل بعد تغييرات.

---

### 🧩 70. كيف تختبر Notifications (push/email/sms)؟

**الجواب:**

- التحقق من الإرسال الفعلي (logs).
- التوقيت والشرط الصحيح.
- التنسيق والمحتوى الصحيح.
- التعامل مع حالات الفشل.

### 🧩 71. كيف تختبر عملية الدفع الإلكترونية؟

**الجواب:**

- تحقق من التكامل مع بوابة الدفع (API response).
- تحقق من البيانات المشفرة (SSL).
- اختبر حالات النجاح، الفشل، الإلغاء، والtimeout.
- تأكد من تسجيل المعاملة بشكل آمن.

---

### 🧩 72. ما الفرق بين Load و Stress Testing؟

**الجواب:**

- **Load Testing**: اختبار الأداء تحت حجم متوقع من المستخدمين.
- **Stress Testing**: اختبار الأداء تحت ضغط يفوق المتوقع لتحديد نقطة الانهيار.

---

### 🧩 73. كيف تُجري اختبار Localization؟

**الجواب:**

- تحقق من الترجمة والمعاني.
- تنسيق التواريخ والأرقام.
- اتجاه النص (مثل RTL للعربية).
- توافق الرموز والألوان ثقافيًا.

---

### 🧩 74. ما الفرق بين Mock و Stub؟

**الجواب:**

- **Stub**: يُعيد بيانات ثابتة مسبقة.
- **Mock**: يتحقق من الطريقة التي تم استدعاؤه بها (parameters, times called...).

---

### 🧩 75. ما هو الفرق بين Testing و Validation و Verification؟

**الجواب:**

- **Testing**: تنفيذ فعلي للحالات.
- **Validation**: هل نُبني الشيء الصحيح؟
- **Verification**: هل نبني الشيء بشكل صحيح؟

---

### 🧩 76. كيف تُجري اختبار Backward Compatibility؟

**الجواب:**

- تشغيل الإصدارات القديمة مع الإصدار الجديد من السيرفر أو القاعدة.
- اختبار حفظ البيانات وقراءتها بالتنسيق القديم.
- فحص API compatibility مع النسخ السابقة من العملاء.

---

### 🧩 77. ما هو مفهوم Shift Left Testing؟

**الجواب:**
نقل نشاط الاختبار إلى مراحل مبكرة في دورة التطوير (Requirements, Design)، لاكتشاف الأخطاء مبكرًا وتقليل التكلفة.

---

### 🧩 78. كيف تختبر التطبيقات التي تعتمد على الزمن (Scheduling, Cron Jobs)؟

**الجواب:**

- استخدام mock للـ time/date.
- اختبار حالات edge مثل منتصف الليل، leap year.
- التأكد من تنفيذ المهام في التوقيتات المحددة.

---

### 🧩 79. ما هو الفرق بين Functional و Non-Functional Testing؟

**الجواب:**

- **Functional**: اختبار ماذا يفعل النظام (Features).
- **Non-Functional**: اختبار كيف يعمل النظام (Performance, Usability, Security).

---

### 🧩 80. ما هي الاختبارات التي يجب أن تُجرى بعد Fix لعيب حرج؟

**الجواب:**

- اختبار Re-test للحالة نفسها.
- Regression للوظائف المرتبطة.
- تحليل الأسباب لتحديد تأثيره.

---

### 🧩 81. ما هو Canary Testing؟

**الجواب:**
نشر الميزات أو النسخة على شريحة صغيرة من المستخدمين لمراقبة تأثيرها قبل التعميم الكامل.

---

### 🧩 82. كيف تختبر تقارير PDF يتم توليدها تلقائيًا؟

**الجواب:**

- التأكد من صحة البيانات والصيغة.
- تحقق من دعم اللغات والتنسيقات.
- استخدام أدوات مقارنة ملفات PDF أو تحليل المحتوى بالنصوص.

---

### 🧩 83. ما هو الفرق بين Test Stub و Test Driver؟

**الجواب:**

- **Stub**: يحاكي وحدة لم تكتمل بعد تُستدعى من قبل الكود.

- **Driver**: كود مؤقت يُستخدم لاختبار وحدة سفلية لا تُستدعى بعد.

---

### 🧩 84. كيف تتعامل مع Bugs غير قابلة لإعادة التكرار (Non-Reproducible Bugs)؟

**الجواب:**

- جمع معلومات النظام (Logs, Version, Time).
- تحليل الظروف المحيطة.
- محاولة تكرارها ببيانات مشابهة أو بيئة التطوير.

---

### 🧩 85. ما الفرق بين Sanity و Regression Testing؟

**الجواب:**

- **Sanity**: اختبار سريع للتأكد أن التغيير لم يُفسد الميزات الرئيسية.
- **Regression**: شامل لكل الميزات للتأكد من أن التحديث لم يؤثر سلبًا.

---

### 🧩 86. ما هو الفرق بين Blocking و Non-blocking Defects؟

**الجواب:**

- **Blocking**: يمنع التقدم في الاختبارات (مثل crash).
- **Non-blocking**: يمكن الاستمرار في الاختبارات بالرغم من وجوده.

---

### 🧩 87. كيف تختبر إمكانية الوصول Accessibility لمستخدم كفيف؟

**الجواب:**

- اختبار باستخدام قارئ شاشة (NVDA, JAWS).
- التأكد من وجود وصف بديل للصور.

- إمكانية التنقل بالكيبورد فقط.
- ترتيب عناصر التركيز (focus order).

---

### 🧩 88. ما هو الفرق بين Test Data و Test Environment؟

**الجواب:**

- **Test Data**: البيانات المُستخدمة خلال الاختبار.
- **Test Environment**: البنية التي يتم فيها تشغيل التطبيق (سيرفر، إعدادات، أدوات).

---

### 🧩 89. ما هو الفرق بين Incident و Defect؟

**الجواب:**

- **Incident**: أي سلوك غير متوقع أثناء الاختبار.
- **Defect**: حادث ثبت أنه ناجم عن خطأ في الكود.

---

### 🧩 90. ما أهمية Coverage Analysis؟

**الجواب:**
تحليل مدى تغطية الاختبارات للكود (branches, paths, conditions)، ويساعد في اكتشاف الأجزاء غير المختبرة.

---

### 🧩 91. ما هو الفرق بين Blocking Bug و Showstopper؟

**الجواب:**
غالبًا يُستخدمان بنفس السياق، لكن:

- **Blocking**: يمنع اختبار ميزة محددة.
- **Showstopper**: يمنع إصدار النظام أو حتى استخدامه.

---

### 🧩 92. ما هو الفرق بين Error Guessing و Exploratory Testing؟

**الجواب:**

- **Error Guessing**: يعتمد على الخبرة لتخمين أماكن الأخطاء الشائعة.
- **Exploratory**: اختبار أثناء التعلم بدون سيناريوهات مسبقة.

---

### 🧩 93. كيف تُعد تقارير اختبار احترافية؟

**الجواب:**

- تغطية عدد الحالات، النجاح، الفشل، النسبة.
- الإشارة إلى العيوب الحرجة.
- الرسوم البيانية، توصيات واضحة.
- مقارنة مع الإصدارات السابقة.

---

### 🧩 94. ما الفرق بين Hard و Soft Assertion في التستنج؟

**الجواب:**

- **Hard**: يوقف الاختبار فور الفشل.
- **Soft**: يسجل الفشل لكن يكمل بقية الاختبار.

---

### 🧩 95. ما هو الفرق بين Regression Suite و Smoke Suite؟

**الجواب:**

- **Smoke Suite**: اختبارات أولية سريعة للوظائف الرئيسية.
- **Regression Suite**: اختبارات شاملة للتأكد من سلامة النظام بعد تغييرات.

---

### 🧩 96. ما أهمية اختبارات الحد الأدنى/الأقصى للمدخلات؟

**الجواب:**
تساعد في اكتشاف أخطاء validation، overflow، أو سوء التعامل مع القيم القصوى والدنيا.

---

## 🧩 97. ما الفرق بين Agile Testing و Traditional Testing؟

**الجواب:**

- **Agile**: تكراري، يعتمد على التعاون، يبدأ مبكرًا.
- **Traditional**: متأخر بعد التطوير، أقل مرونة.

---

## 🧩 98. ما هي أفضل ممارسات اختبار API؟

**الجواب:**

- تحقق من headers, status codes.
- اختبر المدخلات الصحيحة والخاطئة.
- تحقق من الأداء، الأمن، القيود.

---

## 🧩 99. ما الفرق بين Smoke و BVT (Build Verification Test)؟

**الجواب:**
غالبًا يستخدمان بالتبادل، لكن:

- **BVT**: تركّز على استقرار الـ build تلقائيًا بعد كل إصدار.
- **Smoke**: يدوي أو تلقائي للتأكد من الأساسيات قبل البدء في اختبار عميق.

---

## 🧩 100. ما الفرق بين Dead Code و Untested Code؟

**الجواب:**

- **Dead Code**: لا يتم تنفيذه أبدًا، موجود لكنه غير مستخدم.
- **Untested Code**: موجود ويُستخدم، لكن لم يُختبر بعد.

### 1. كيف تحدد أولويات حالات الاختبار في بيئة بها وقت وموارد محدودة؟

✅ الإجابة:
أستخدم **تحليل المخاطر (Risk-Based Testing)** لتحديد الاختبارات الأكثر تأثيرًا على العمل، وأعطي الأولوية لوظائف الـCore، والتكاملات الحرجة، ثم السيناريوهات ذات الاحتمالية العالية للفشل.

---

### 2. كيف تتعامل مع تضارب بين المطور وفريق التستنج حول أولوية bug؟

✅ الإجابة:
أستند إلى تأثير العيب على العمل (Business Impact) بدلًا من الرأي الفني فقط. أعرض حالات الاستخدام، وتحليلات العملاء أو الإنتاج لدعم حجتي.

---

### 3. كيف تصمم خطة اختبار لنظام جديد ومعقد؟

✅ الإجابة:
أبدأ بتحليل المتطلبات والسيناريوهات المعقدة، ثم أبني خطة تشمل:

- نطاق التغطية.
- الأدوات.
- المسؤوليات.
- جدول زمني.
- خطة اختبار للأداء، الأمان، والـRegression.

---

### 4. كيف تبني Test Strategy لمنتج SaaS؟

✅ الإجابة:
أغطي الجوانب التالية:

- Coverage متعددة المتصفحات والأجهزة.
- اختبارات CI/CD وDeployment Verification.
- اختبارات Security وData privacy.
- Monitoring للـProduction.

---

### 5. كيف تختار أدوات التستنج؟

✅ الإجابة:
أُقيّم الأدوات بناءً على:

- توافقها مع tech stack.
- سهولة الدمج في CI/CD.
- قابلية التوسعة.
- تكلفة الصيانة.
- دعم المجتمع/الوثائق.

---

### 6. ماذا تفعل لو طُلب منك التسليم بدون إنهاء التستنج؟

✅ الإجابة:
أُوضح المخاطر بتقرير واضح، أُحدد ما تم اختباره وما لم يُختبر، وأقترح بدائل مثل إطلاق جزئي أو تمديد للموعد.

---

### 7. ما معاييرك لتقرير إن كان المنتج جاهزًا للإطلاق؟

✅ الإجابة:

- عدد العيوب المفتوحة.
- نسبة اجتياز الاختبارات الحرجة.
- نتائج performance/security tests.
- موافقة العميل أو PO.
- تحليل المخاطر المتبقية.

---

### 8. كيف تقيس فعالية فريق التستنج؟

✅ الإجابة:
باستخدام مؤشرات مثل:

- Defect Leakage.

- Test Case Effectiveness.
- Test Execution Rate.
- Time to Detect/Resolve Defects.

---

### 9. كيف تتعامل مع فريق تطوير لا يحترم التستنج؟

✅ الإجابة:
أبني الثقة من خلال التعاون المبكر، مشاركة تقارير مؤثرة، والمشاركة في الـDaily Standups. أثبت فعالية الفريق بنتائج ملموسة.

---

### 10. احكِ عن تجربة أدرت فيها فريق Testing متعدد الثقافات.

✅ الإجابة:
اعتمدت على توحيد العمليات، أدوات مشتركة، مراجعات جماعية للـTest Cases، وتشجيع التواصل المفتوح لتجاوز الفجوات الثقافية واللغوية.

---

## ⚙️ أسئلة تقنية واختبارات متقدمة (11–30)

---

### 11. كيف تختبر Microservices Architecture؟

✅ الإجابة:
باستخدام اختبارات:

- Contract Testing (Pact).
- Integration Testing عبر الـAPIs.
- Fault Tolerance (Chaos Testing).
- Performance لكل خدمة.

---

### 12. كيف تختبر REST APIs؟

✅ الإجابة:
أجري اختبارات:

- Functional (status, headers, payloads).
- Security (Auth, SQLi).
- Load.
- Error Handling (timeouts, malformed requests).

---

### 13. كيف تصمم Automation Framework من الصفر؟

✅ الإجابة:
أحدد:

- اللغات (مثلاً Python+PyTest).
- الهيكل (Page Object Model).
- إدارة البيانات.
- تقارير التنفيذ.
- قابلية التوسع.
- الدمج مع CI/CD.

---

### 14. ما الفرق بين Test Harness و Test Framework؟

✅ الإجابة:

- **Harness**: أدوات ومكتبات لتشغيل الاختبارات وتقييم النتائج.
- **Framework**: بنية متكاملة تشمل تنظيم الأكواد، التقارير، إعادة الاستخدام.

---

### 15. كيف تتأكد من استقرار اختبارات الـUI؟

✅ الإجابة:

- استخدام Waits الذكية.

- تجنب الاعتماد على العناصر المتغيرة.
- تنظيف البيئة قبل كل run.
- Parallelization بحذر.

---

### 16. كيف تختبر نظام دفع؟

✅ الإجابة:
أُجري اختبارات:

- تكامل Gateway.
- Fraud Detection.
- Security (TLS, PCI compliance).
- سيناريوهات Edge (double submit, cancel...).

---

### 17. كيف تختبر Caching؟

✅ الإجابة:

- التحقق من سرعة الاسترجاع.
- تغيّر البيانات بعد invalidation.
- مقارنة نتائج cached vs uncached.
- اختبار الـTTL، التزامن.

---

### 18. ما الفرق في التستنج بين Monolith و Microservices؟

✅ الإجابة:

- Microservices تتطلب اختبارًا لعقود البيانات، التوازي، الفشل الجزئي.
- Monolith يسهل التغطية الشاملة ولكنه أقل مرونة.

---

### 19. كيف تموّك dependency في اختبارات وحدات؟

✅ الإجابة:
باستخدام مكتبات مثل `Mockito`, `unittest.mock` لمحاكاة الردود وسلوك الخدمات الخارجية.

---

### 20. كيف تختبر Event-driven Systems؟

✅ الإجابة:

- محاكاة الأحداث.
- التأكد من معالجة الحدث بالترتيب.
- اختبار الـDead-letter queues.
- Monitoring لـ async errors.

### 21. كيف تصمم اختبار تحميل لنظام e-commerce؟

✅ الإجابة:
أُحدد السيناريوهات الرئيسية (تصفح، شراء، دفع)، أستخدم أدوات مثل JMeter أو Gatling، وأحاكي حمل المستخدمين تدريجيًا مع مراقبة استهلاك الموارد والاستجابة.

---

### 22. ما الأدوات التي استخدمتها لتحليل الأداء؟ ولماذا؟

✅ الإجابة:

- **JMeter** لتحليل التحمل.
- **Grafana + Prometheus** للمراقبة الحية.
- **Lighthouse** لتحليل واجهة الويب.
  اخترتها لأنها مفتوحة المصدر، مرنة، وتتكامل بسهولة مع CI.

---

### 23. كيف تكتشف memory leak؟

✅ الإجابة:

- مراقبة GC logs.
- أدوات مثل VisualVM, Valgrind.
- مراقبة RAM أثناء التشغيل المستمر.
- تحليل heap dump وتكرار السلوكيات.

---

### 24. كيف تختبر النظام عند انقطاع مفاجئ للاتصال؟

✅ الإجابة:

- اختبارات فشل الشبكة (network throttling tools).
- محاكاة فقدان الاتصال والعودة.
- مراقبة حفظ الحالة (state), وإعادة المحاولة (retry logic).

---

### 25. كيف تتعامل مع flaky tests؟

✅ الإجابة:

- التحقيق في السبب (توقيت؟ بيانات؟ سباق حالات؟).
- تثبيت الـwaits.
- إعادة كتابة الاختبار بتصميم أفضل.
- عزلها عن الـpipeline حتى إصلاحها.

---

### 26. كيف تحدد Bottlenecks باستخدام البيانات فقط؟

✅ الإجابة:

- تحليل Latency في الـlogs.
- مراقبة CPU, RAM, DB calls.
- تحليل time to process لكل endpoint أو وظيفة.

---

### 27. كيف تكتب سيناريوهات performance واقعية؟

✅ الإجابة:

- تحليل سلوك المستخدم من الإنتاج.
- تحديد الأنماط والـpeaks.
- محاكاة سيناريوهات التصفح/البحث/الشراء بنسبة حقيقية.

---

### 28. كيف تختبر سلوك النظام بموارد منخفضة؟

✅ الإجابة:

- تشغيله في حاويات بحدود RAM/CPU.
- مراقبة الأداء والانهيارات.
- فحص graceful degradation.

---

### 29. ما الفرق بين throughput و latency؟

✅ الإجابة:

- **Latency:** وقت الاستجابة الفردية.
- **Throughput:** عدد الطلبات التي يعالجها النظام في فترة زمنية.

---

### 30. كيف تحلل نتائج اختبار الضغط وتكتب تقريرًا تنفيذيًا؟

✅ الإجابة:

- إبراز response time, error rate, resource use.
- تحديد النقاط الحرجة.
- تقديم توصيات فنية قابلة للتنفيذ (scaling, caching...).

---

## 🔄 أسئلة CI/CD و(31–40) (Automation)

---

### 31. كيف تدمج اختباراتك مع CI/CD؟

✅ الإجابة:

- ربط الاختبارات بالـGit Hooks أو Jenkins Pipelines.
- التقسيم إلى stages (unit > integration > UI).
- نشر النتائج وتقارير failures تلقائيًا.

---

### 32. ما تجربتك مع أدوات CI مثل Jenkins / GitHub Actions؟

✅ الإجابة:

- إعداد بيئات Build/Test.
- Configurations بالـYAML.
- إعداد triggers حسب نوع الفرع أو التغيير.

---

### 33. كيف تدير test data في بيئات أوتوماتيكية؟

✅ الإجابة:

- استخدام Fixtures أو Mocked data.
- عزل البيانات لكل run.
- تنظيف البيئة بعد كل تنفيذ.

---

### 34. كيف تتعامل مع flaky tests في مرحلة build؟

✅ الإجابة:

- عزلها مؤقتًا.
- التحقيق في الأسباب.
- تحديد Retry logic مع شروط صارمة.

---

### 35. كيف توازن بين التستنج اليدوي والتلقائي؟

✅ الإجابة:

- التلقائي للمهمات المتكررة أو الـRegression.
- اليدوي للـUX, Exploratory, والعناصر المتغيرة كثيرًا.

---

### 36. متى تستخدم E2E Testing؟

✅ الإجابة:

- عندما أحتاج التحقق من تدفق كامل للمستخدم.
- عند الربط بين أنظمة مختلفة.
- نادرًا، بسبب التكلفة والبطء.

---

### 37. ما أفضل ممارسات كتابة اختبارات قابلة للصيانة؟

✅ الإجابة:

- Page Object Pattern.
- DRY principles.
- أسماء وصفية.
- تجنب البيانات الثابتة داخل الأكواد.

---

### 38. كيف تقلل وقت اختبارات Regression؟

✅ الإجابة:

- الاختبارات التزايدية.
- تنفيذ متوازي.
- اختيار الاختبارات بناءً على التغييرات في الكود.

---

### 39. كيف تقيس تغطية الاختبارات بذكاء؟

✅ الإجابة:

- استخدام أدوات coverage مع تحليل السياق.
- التركيز على الأكواد المعقدة والحرجة.
- عدم السعي وراء 100% بل تغطية "ذات قيمة".

---

### 40. ما الفرق بين smoke وsanity tests؟

✅ الإجابة:

- **Smoke:** اختبارات شاملة لأهم الوظائف بعد كل build.
- **Sanity:** اختبارات مركزة على التعديلات الأخيرة فقط.

---

## 🧩 أسئلة فكر نقدي وتحليل (41–50)

---

### 41. كيف تحقق في bug في الإنتاج؟

✅ الإجابة:

- قراءة الـlogs.
- إعادة إنتاج الخطأ.
- فحص الكود المرتبط.

- الرجوع للـcommit المسبب.
- تقديم RCA Report.

---

### 42. كيف تختبر بدون مستندات واضحة؟

✅ الإجابة:

- التواصل مع الـPO/المطورين.
- تحليل الـUI و الـAPI.
- تطبيق exploratory testing.
- توثيق الفرضيات.

---

### 43. اذكر موقفًا استخدمت فيه Exploratory Testing وكان فعالًا.

✅ الإجابة:
في مشروع بدون مستندات UI، اكتشفت عدة عيوب usability وnavigation من خلال session-based exploratory testing.

---

### 44. كيف تقنع الإدارة بوقت إضافي للاختبارات؟

✅ الإجابة:

- عرض bugs التي تم تفاديها سابقًا.
- مقارنة بين تكلفة إصلاح defect قبل/بعد الإنتاج.
- تقديم خطة واضحة لاستخدام الوقت الإضافي.

---

### 45. ما المقياس لتقييم جودة Test Case؟

✅ الإجابة:

- وضوح الخطوات.

- إمكانية إعادة استخدامها.
- قابليتها للأتمتة.
- تغطيتها للمخاطر.

---

### 46. متى تتوقف عن صيانة اختبار تلقائي؟

✅ الإجابة:

- إذا أصبح مكلفًا أكثر من فائدته.
- إذا تغيّر النظام جذريًا.
- إذا تم دمج الاختبار في نظام آخر.

---

### 47. كيف تختبر في Agile سريع؟

✅ الإجابة:

- Shift-left testing.
- مراجعة القصص فور كتابتها.
- Automation سريع وفعال.
- تقسيم الجهد عبر الفريق.

---

### 48. كيف تبني علاقة فعالة مع المطورين؟

✅ الإجابة:

- مراجعات الكود معًا.
- فهم الكود.
- استخدام لغة غير تصادمية.
- الاحتفال بالنجاحات المشتركة.

---

### 49. كيف تختبر منتج مبني على الذكاء الاصطناعي؟

✅ الإجابة:

- التركيز على accuracy, bias, ونتائج غير متوقعة.
- اختبار البيانات والتدريب.
- مقارنة المخرجات بتوقعات البشر.

---

### 50. دخلت مشروعًا في منتصفه، ما أول 3 أشياء تفعلها؟

✅ الإجابة:

1. قراءة الوثائق والـBacklog.
2. استكشاف المنتج عمليًا.
3. مراجعة الحوارات والتقارير السابقة لفهم الفجوات.