الأكاديمية العربية الدولية
Arab International Academy

# هندسة البرمجيات

# Software Engineering

---

عبدالقادر العبدالله

كلية العلوم – تخصص البرمجة

# المحاور

1. المقدمة
2. أهداف هندسة البرمجيات
3. دورة حياة البرمجيات(SDLC)
4. أهمية تحليل المتطلبات
5. النماذج المستخدمة في هندسة البرمجيات
6. التصميم البرمجي
7. كتابة الشيفرة البرمجية
8. الاختبار وضمان الجودة
9. التوثيق في هندسة البرمجيات
10. الصيانة البرمجية
11. النموذج الشلالي(Waterfall Model)
12. منهجية الرشيقة(Agile)
13. النماذج التكرارية(Iterative Models)
14. هندسة المتطلبات(Requirements Engineering)
15. البرمجيات كمنتج(Software as a Product)
16. التحديات في تطوير البرمجيات
17. نماذج التصميم البرمجي(Design Models)
18. جودة البرمجيات(Software Quality)
19. الصيانة الوقائية(Preventive Maintenance)
20. إدارة المشاريع البرمجية(Software Project Management)
21. الاختبار الآلي(Automated Testing)
22. البرمجيات المفتوحة المصدر(Open Source Software)
23. البرمجة كائنية التوجه(Object-Oriented Programming)
24. الأمان البرمجي(Software Security)
25. الحوسبة السحابية(Cloud Computing)
26. البرمجيات كخدمة(SaaS)
27. هندسة الأنظمة المدمجة(Embedded Systems)
28. دور الذكاء الاصطناعي في هندسة البرمجيات
29. هندسة البرمجيات التكيفية(Adaptive Software Engineering)
30. تطوير واجهات المستخدم(User Interface Development)
31. البرمجيات والبيانات الضخمة(Big Data)
32. التوجهات المستقبلية في هندسة البرمجيات

# المخرجات المتوقعة من الدرس

قدرة الطالب على تعريف هندسة البرمجيات وإدراك مفاهيمها

قدرة الطالب على تعريف مراحل بناء النظام أو التطبيق البرمجي

معرفة الطالب بالنماذج المختلفة لتطوير البرمجيات وقدرته على التمييز بينها

التعرف على بعض الادوات المساعدة على بناء البرمجيات وتطويرها

# المقدمة

**تعريف هندسة البرمجيات** :مجال يهدف إلى تصميم وتطوير الأنظمة البرمجية بجودة عالية حيث تهدف هذه الانظمة إلى تلبية إحتياجات شريحة او فئة من المستخدمين عن طريق أداء مهمة واحدة او عدة مهمام ضمن نفس التطبيق
نطلق على العلم الذي يدرس بناء هذه الانظمة والتطبيقات من بداية تخطيط المشروع وحتى إطلاقه وصيانته هندسة البرمجيات.

**أهمية هندسة البرمجيات** :تلبية الاحتياجات المتزايدة لتطبيقات تقنية فعّالة وقابلة للصيانة.

**تحديات المجال:**

زيادة التعقيد البرمجي.
الحفاظ على الجودة العالية مع قيود الوقت.
معالجة المشكلات الأمنية.

ومن الممكن التعبير عن هندسة البرمجيات بدورة حياة تطوير التطبيقات البرمجية إنطلاقا من مرحلة دراسة المشروع وتحليل متطلباته وصلولا إلى تنفيذه وإختباره وتشمل ايضاً مرحلة الصيانة والتحديث

# أهداف هندسة البرمجيات

**إنتاج برمجيات موثوقة** : تعمل بكفاءة دون أخطاء.

**توفير قابلية الصيانة** : القدرة على تحديث البرمجيات بمرونة.

**تحقيق الاستدامة** : برامج تدوم لفترات طويلة.

**زيادة الإنتاجية** : تحسين وقت التنفيذ والتسليم .

# دورة حياة البرمجيات (SDLC)

هي عملية تطوير البرمجيات عبر مراحل منظمة.

**المراحل الرئيسية:**
التخطيط: تحديد الأهداف والموارد.
التحليل: جمع وفهم المتطلبات.
التصميم: وضع الهيكلية التقنية.
التطوير: كتابة الشيفرة البرمجية.
الاختبار: ضمان الجودة والأداء.
الصيانة: تحديث النظام بعد الإطلاق.

قد يتم الإنتقال بين المراحل بعد إجراء الإختبارات على البرمجيات للتحقق من أداء المهام المطلوبة وتطويرها حسب الحاجة

# أهمية تحليل المتطلبات

المتطلبات هي وصف لما يجب أن يفعله النظام وتحديد المهام التي من المقرر أن تنفذها البرمجيات المضمنة في هذا النظام .

**أنواع المتطلبات:**

- **وظيفية** : ما يجب أن يفعله النظام.
- **غير وظيفية** :القيود (مثل الأداء والأمان).

**أهداف التحليل:**

- تحسين فهم الاحتياجات.
- تقليل التعديلات المكلفة لاحقًا.
- تعزيز رضا العميل.

# النماذج المستخدمة في هندسة البرمجيات

النماذج هي منهجيات معينة في تسيير دورة حياة البرمجيات بناءً على لوائح وترتيبات معينة تندرج تحت الأسماء التالية:

**النموذج الشلالي:(Waterfall)**

- يتم العمل فيه بشكل متسلسل.
- مناسب للمشاريع ذات المتطلبات الثابتة.

**النموذج التكراري:(Iterative)**

- يعتمد على تحسينات متكررة.
- مناسب للمشاريع الديناميكية.

**المنهجية الرشيقة:(Agile)**

- التركيز على التعاون والاستجابة للتغيرات.

# التصميم البرمجي

هو عملية تحديد البنية الهيكلية للنظام حيث تلعب هذه العملية دور مهم جدا في فهم آلية عمل النظام وترابط المهام المختلفة في ما بينها وقد يحتاج النظام إلى إعادة التصميم عدة مرات بناءً على التغيرات الطارئة على المشروع.

**عناصر التصميم:**

- الهيكلية العامة.(Architecture)
- التصميم التفصيلي.(Detailed Design)

**أهداف التصميم:**

- تحسين الأداء.
- ضمان قابلية التوسع.

# كتابة الشيفرة البرمجية

وهي مرحلة كتابة الرمز (الكود البرمجي) بإستخدام لغة برمجة معينة يتم تحديدها بناءً على تحليل متطلبات المشروع والتصميم وغيرها .
قد تعتبر هذا المرحلة هي الأكثر شهرةً بين مراحل بناء التطبيقات والأنظمة.

**أفضل الممارسات:**
استخدام تسميات واضحة للمتغيرات.
كتابة تعليقات توضيحية.
التقيد بالمعايير القياسية.
**أدوات الدعم:**
IDEs مثل .(Visual Studio, Eclipse)
أدوات التحكم بالإصدارات مثل(Git)

# الاختبار وضمان الجودة

مرحلة الإختبار تعتبر من المراحل المهمة في دورة حياة تطوير التطبيقات البرمجية بحيث تساعد المطورين على إكتشاف الأخطاء التي من الوارد حدوثها أثناء إستخدام التطبيق

هذه المرحلة بالغة الأهمية خصوصا في التطبيقات والأنظمة الموجهة للقطاع الطبي والأمني والصناعي .

**أنواع الاختبار:**

- اختبار الوحدة :(Unit Testing) لاختبار أجزاء صغيرة.
- اختبار التكامل :(Integration Testing) لاختبار توافق الوحدات معًا.
- اختبار النظام :(System Testing) لاختبار النظام بالكامل.

**أهداف الاختبار:**

- كشف الأخطاء.
- تحسين موثوقية النظام.

# التوثيق في هندسة البرمجيات

التوثيق هو كتابة مستندات توضح النظام واجزائه وآليات عمله.
يعمل على بناء التوثيق مجموعة المبرمجين والمحللين الذين قاموا ببناء المشروع وتطويره ويهدف إلى تسهيل عمليات التطوير والصيانة وترتيب المفاهيم اللازمة لإضافة ميزات جديدة في المستقبل كما يساعد المستخدمين على إستخدام النظام بشكل صحيح وفعال.

**أنواع التوثيق:**

- التوثيق التقني (للبرمجيين).
- التوثيق المستخدم (للمستخدمين النهائيين).

**أهمية التوثيق:**

- تسهيل الصيانة.
- ضمان فهم النظام من قبل الفريق.

# الصيانة البرمجية

تركز هذه المرحلة على تصحيح الأخطاء الناتجة أثناء كتابة الشيفرة المصدرية أو اثناء الإستخدام الخاطئ للنظام أو التطبيق البرمجي.

قد تحتاج بعض الأنظمة إلى عمليات صيانة دورية لضمان إستمرارية عملها بالشكل المطلوب بغض النظر عن حدوث خلل فني مباشر

**أنواع الصيانة:**

- تصحيح الأخطاء.
- تحسين الأداء.
- إضافة ميزات جديدة.

**التحديات:**

- صعوبة فهم الشيفرة القديمة.
- التكاليف المرتفعة للصيانة.

**أهمية التخطيط للصيانة أثناء التطوير:**

- تقليل المشاكل المستقبلية.

# النموذج الشلالي (Waterfall Model)

نموذج خطي يتم فيه إتمام كل مرحلة قبل الانتقال إلى المرحلة التالية.

**المراحل الأساسية:**

- جمع وتحليل المتطلبات.
- التصميم.
- التنفيذ (التطوير).
- الاختبار.
- التسليم والصيانة.

**مزايا النموذج الشلالي:**

- مناسب للمشاريع ذات المتطلبات الثابتة.
- يوفر وضوحًا في التقدم بين المراحل.

**العيوب:**

- غير مرن للتغييرات.
- قد يؤدي إلى مشاكل إذا تم اكتشاف أخطاء في المراحل المتأخرة.

# منهجية Agile (الرشيقة)

أسلوب تطوير مرن يركز على التعاون بين الفريق والاستجابة السريعة للتغيرات.

**الركائز الأساسية:**

- الأفراد والتفاعل على العمليات والأدوات.
- البرمجيات الجاهزة على التوثيق الشامل.
- التعاون مع العميل على التفاوض.
- الاستجابة للتغيير على الالتزام بالخطة.

**فوائد منهجية:Agile**

- تسريع زمن التسليم.
- تحسين رضا العملاء.
- تعزيز التعاون داخل الفريق.

**التحديات:**

- يتطلب فريقًا متكاملاً وملتزمًا.
- قد يواجه صعوبة في المشاريع الكبيرة.

# النماذج التكرارية (Iterative Models)

عملية تطوير متكررة يتم فيها بناء النظام على شكل نسخ متتابعة.

**العملية:**

- تحديد المتطلبات الأساسية.
- تصميم نسخة مبدئية.
- تنفيذ النسخة واختبارها.
- تحسين النسخة وإضافة ميزات جديدة.

**فوائد النماذج التكرارية:**

- تقليل المخاطر من خلال تحسينات مستمرة.
- إعطاء العملاء فرصة لتقييم النسخ الأولية.

**أمثلة على التطبيقات :**

- مشاريع التطبيقات الكبيرة والمعقدة .

# هندسة المتطلبات (Requirements Engineering)

هي عملية جمع وتحليل وتوثيق متطلبات العميل أو المستخدم للنظام.

**خطوات هندسة المتطلبات:**

- **جمع المتطلبات** :باستخدام المقابلات، الاستبيانات، والملاحظات.
- **تحليل المتطلبات** :لفهم مدى قابلية تنفيذها.
- **توثيق المتطلبات** :إنشاء مستند واضح وشامل.
- **التحقق من المتطلبات** :لضمان تلبيتها لاحتياجات العميل.

**أهميتها:**

- تحسين جودة البرمجيات النهائية.
- تقليل المخاطر الناتجة عن سوء الفهم.

# كويز -1

1. ما هو الهدف الأساسي من هندسة البرمجيات؟
2. ما هي المرحلة الأولى في دورة حياة البرمجيات(SDLC) ؟
3. اذكر فائدتين لتحليل المتطلبات الجيد.
4. ما هو الفرق الأساسي بين النموذج الشلالي والنموذج التكراري؟
5. اذكر النموذج البرمجي الذي يتميز بالمرونة والاستجابة السريعة للتغيرات

# الأجوبة - 1

1. إنتاج برمجيات تلبي احتياجات المستخدم وتكون قابلة للصيانة.

2. التخطيط.

3. تحسين فهم احتياجات العميل, تقليل التعديلات المكلفة لاحقًا .

4. النموذج الشلالي يعتمد على إتمام كل مرحلة قبل الانتقال إلى المرحلة التالية , النموذج التكراري يسمح بتكرار المراحل لتحسين النظام .

5. منهجية Agile.

# البرمجيات كمنتج (Software as a Product)

البرمجيات التي يتم تطويرها لتلبية احتياجات مستخدمين محددين أو أسواق معينة.

**خصائص البرمجيات كمنتج:**

- قابلة للاستخدام عبر الزمن.(Sustainability)
- سهولة التحديث والتخصيص.
- تقديم قيمة مضافة للمستخدمين.

**أنواع المنتجات البرمجية:**

- تطبيقات الأجهزة المحمولة.
- أنظمة إدارة البيانات.
- الأدوات البرمجية المتخصصة.

**أمثلة شهيرة:**

- تطبيقات Office.
- أنظمة تخطيط موارد المؤسسات.(ERP)

# التحديات في تطوير البرمجيات

**التحديات التقنية:**

- التعامل مع الأنظمة القديمة.
- التوافق مع الأجهزة المختلفة.
- تحسين الأداء والكفاءة.

**التحديات الإدارية:**

- إدارة فرق العمل الكبيرة.
- الالتزام بالميزانية والجدول الزمني.

**التحديات الأمنية:**

- الحماية من الهجمات الإلكترونية.
- ضمان سرية البيانات.

# نماذج التصميم البرمجي (Design Models)

مخططات تُستخدم لتنظيم وتوضيح بنية النظام.

**أنواع نماذج التصميم:**

- النماذج الهيكلية (Structural Models): تحدد العلاقات بين المكونات.
- النماذج الديناميكية (Dynamic Models): تركز على سلوك النظام بمرور الوقت.
- نماذج الكائنات (Object-Oriented Models): تستخدم الكائنات في التصميم.

**فوائد النماذج:**

- تحسين التفاهم بين الفرق.
- تقليل الأخطاء أثناء التطوير.

# جودة البرمجيات (Software Quality)

القدرة على تقديم نظام يلبي متطلبات العميل ويعمل بشكل موثوق.

**أبعاد الجودة:**

- الأداء.
- القابلية للاستخدام.
- القابلية للصيانة.
- الأمان.

**أدوات تقييم الجودة:**

- أدوات اختبار الأداء.
- مراجعة الأكواد.
- مقاييس الجودة مثل ISO 9126.

# الصيانة الوقائية (Preventive Maintenance)

هي تحسين البرمجيات لتجنب المشكلات المستقبلية.

**أمثلة:**

- تحديث مكونات النظام لدعم تقنيات جديدة.
- تحسين الأداء لمعالجة النمو في عدد المستخدمين.

**فوائد الصيانة الوقائية:**

- تقليل تكلفة المشكلات المستقبلية.
- تعزيز استدامة النظام.
- تحسين تجربة المستخدم.

# إدارة المشاريع البرمجية (Software Project Management)

عملية تنظيم وتخطيط الموارد والأنشطة لتحقيق أهداف المشروع البرمجي.

**العناصر الأساسية لإدارة المشاريع البرمجية:**

- **التخطيط** :وضع الجداول الزمنية وتخصيص الموارد.
- **التنفيذ** :التأكد من سير العمل وفق الخطة.
- **المراقبة والتحكم** :تتبع التقدم ومعالجة الانحرافات.
- **الإغلاق** :تسليم المشروع النهائي وتوثيق العمليات.

**أدوات إدارة المشاريع:**

- برامج مثل Microsoft Project و.Jira
- أدوات التعاون مثل.Trello

**التحديات:**

- إدارة التغيير.
- الحفاظ على الجودة والالتزام بالميزانية.

# الاختبار الآلي (Automated Testing)

استخدام الأدوات البرمجية لاختبار النظام تلقائيًا بدلاً من الاختبار اليدوي.

**فوائد الاختبار الآلي:**

- تسريع عملية الاختبار.
- تحسين دقة النتائج.
- إمكانية تكرار الاختبارات بسهولة.

**أدوات شائعة للاختبار الآلي:**

- Selenium لاختبار الويب.
- JUnitلاختبارات Java.

**أنواع الاختبارات الآلية:**

- اختبارات الوحدة.
- اختبارات الأداء.

# البرمجيات المفتوحة المصدر (Open Source Software)

برامج يتم توفير كودها المصدر علنًا، مما يسمح للمطورين بالمساهمة أو التعديل عليها.

**أمثلة شهيرة:**

- نظام التشغيل Linux.
- مكتبة TensorFlow للتعلم الآلي.

**مزايا البرمجيات المفتوحة المصدر:**

- تكاليف منخفضة.
- مجتمع دعم قوي.
- تحسين مستمر من خلال مساهمات المجتمع.

**التحديات:**

- نقص الدعم الاحترافي.
- تعقيد التعديلات في المشاريع الكبيرة.

# البرمجة كائنية التوجه (Object-Oriented Programming – OOP)

نموذج برمجي يعتمد على الكائنات التي تجمع بين البيانات والوظائف.

**المبادئ الأساسية:**

- **التغليف (Encapsulation)**: حماية البيانات من التلاعب المباشر.
- **الوراثة (Inheritance)**: إعادة استخدام الأكواد بين الكائنات.
- **التعددية (Polymorphism)**: استخدام نفس الطريقة بطرق متعددة.
- **التجريد (Abstraction)**: إخفاء التفاصيل المعقدة وتقديم الوظائف الأساسية فقط.

**أمثلة على OOP لغات مثل:**

- Java
- Python
- C++

# كويز - 2

1. ما المقصود بالبرمجيات كمنتج(Software as a Product) ؟
2. ما هي التحديات الأمنية في تطوير البرمجيات؟
3. ما هو الهدف الأساسي من الصيانة الوقائية؟
4. اذكر مثالين على البرمجيات المفتوحة المصدر.
5. ما هي المبادئ الأربعة الأساسية للبرمجة كائنية التوجه(OOP) ؟

# 2- الأجوبة

1. برامج تُصمم لتلبية احتياجات محددة وتكون قابلة للتحديث والصيانة بمرور الوقت.
2. الحماية من الهجمات السيبرانية, ضمان أمان البيانات, التعامل مع الثغرات الأمنية.
3. تقليل المشكلات المستقبلية وتحسين استدامة النظام.
4. TensorFlow , Linux.
5. التغليف(Encapsulation) ,الوراثة(Inheritance) ,التعددية(Polymorphism) ,التجريد(Abstraction)

# الأمان البرمجي (Software Security)

حماية الأنظمة البرمجية من التهديدات والهجمات.

**أهم التهديدات:**

- الاختراقات.(Hacking)
- البرمجيات الخبيثة.(Malware)
- هجمات الحرمان من الخدمة.(DDoS)

**أساليب تحسين الأمان:**

- التحقق من صحة البيانات المدخلة.
- استخدام التشفير لحماية البيانات.
- تحديث البرمجيات باستمرار لسد الثغرات الأمنية.

**أدوات الأمان:**

- برامج مكافحة الفيروسات.
- جدران الحماية.(Firewalls)

# الحوسبة السحابية (Cloud Computing) في هندسة البرمجيات

استخدام الإنترنت لتقديم خدمات الحوسبة مثل التخزين وقوة المعالجة.

**أمثلة على خدمات السحابة:**

- التخزين: Google Drive، Dropbox .
- المعالجة: Amazon Web Services (AWS)

**فوائد الحوسبة السحابية:**

- تقليل التكاليف التشغيلية.
- زيادة قابلية التوسع.
- تحسين التعاون.

**التحديات:**

- الأمان وحماية البيانات.
- الاعتماد على الاتصال بالإنترنت.

# البرمجيات كخدمة (SaaS)

نموذج يتيح للمستخدمين الوصول إلى التطبيقات عبر الإنترنت بدلاً من تثبيتها محليًا.

**أمثلة شائعة:**

- Google Workspace (Docs, Sheets).
- Microsoft 365.

**مزايا:SaaS**

- سهولة الوصول من أي مكان.
- التحديثات التلقائية.
- تقليل التكاليف للمستخدم النهائي.

**عيوب:SaaS**

- قضايا الأمان والخصوصية.
- اعتماد كبير على الإنترنت.

# هندسة الأنظمة المدمجة (Embedded Systems)

الأنظمة التي تجمع بين البرمجيات والأجهزة لتحقيق مهام محددة.

**أمثلة على الأنظمة المدمجة:**

- أنظمة السيارات.
- الأجهزة المنزلية الذكية.

**خصائص الأنظمة المدمجة:**

- استهلاك منخفض للطاقة.
- أداء عالي لمهام محددة.

**أهمية هندسة البرمجيات فيها:**

- تصميم البرمجيات لتحسين أداء الأجهزة.
- ضمان التوافق بين المكونات المادية والبرمجية.

# دور الذكاء الاصطناعي (AI) في هندسة البرمجيات

استخدام تقنيات الذكاء الاصطناعي لتحسين تطوير البرمجيات .

**تطبيقات AI في هندسة البرمجيات:**

- تحليل المتطلبات باستخدام معالجة اللغة الطبيعية.(NLP)
- تحسين الشيفرة البرمجية باستخدام التعلم الآلي.(ML)
- تحسين الاختبار واكتشاف الأخطاء تلقائيًا.

**أدوات شائعة:**

- GitHub Copilot.
- أدوات التنبؤ بتحليل البيانات.

# هندسة البرمجيات التكيفية (Adaptive Software Engineering)

عملية تطوير برمجيات قادرة على التكيف مع التغيرات البيئية أو متطلبات المستخدم.

**خصائص البرمجيات التكيفية:**

- القدرة على التعلم من البيانات الجديدة.
- التفاعل مع المستخدم لتخصيص التجربة.
- التكيف مع التغيرات التكنولوجية.

**أمثلة على الأنظمة التكيفية:**

- أنظمة التوصية) مثل.(Netflix
- تطبيقات التجارة الإلكترونية التي تتكيف مع سلوك المستخدم.

**فوائد الهندسة التكيفية:**

- تحسين رضا المستخدم.
- استمرارية الأداء في ظروف متغيرة.

# تطوير واجهات المستخدم (User Interface Development)

عملية تصميم وتطوير عناصر التفاعل بين المستخدم والنظام.

**مبادئ التصميم الجيد:**

- البساطة وسهولة الاستخدام.
- التناسق في التصميم.
- استجابة سريعة لإجراءات المستخدم.

**أدوات التصميم:**

- Figma، Adobe XD لتصميم الواجهات.
- Bootstrapو Material UI لتطويرها.

**اختبار واجهات المستخدم:**

- جمع ملاحظات المستخدمين.
- تحسين تجربة الاستخدام بناءً على ردود الفعل.

# البرمجيات والبيانات الضخمة (Big Data)

**البيانات الضخمة** :مجموعات بيانات هائلة لا يمكن تحليلها باستخدام الطرق التقليدية.

**دور البرمجيات في إدارة البيانات الضخمة:**

جمع البيانات ومعالجتها.

تحليل البيانات لاتخاذ قرارات أفضل.

تخزين البيانات باستخدام أنظمة تخزين متقدمة.

**أمثلة على تطبيقات البيانات الضخمة:**

تحسين عمليات التسويق.

تحليل الاتجاهات في وسائل التواصل الاجتماعي.

# التوجهات المستقبلية في هندسة البرمجيات

**تطور التقنيات:**

- زيادة استخدام الذكاء الاصطناعي.
- تعزيز تطبيقات البلوك تشين.
- نمو تقنيات الواقع الافتراضي (VR) والواقع المعزز.(AR)

**التوجه نحو البرمجيات المستدامة** :تقليل استهلاك الطاقة والموارد.
**زيادة أهمية الأمان السيبراني** :مواكبة التهديدات الجديدة.
**التكامل بين البرمجيات والبيانات** :استخدام البيانات الضخمة لتحسين البرمجيات .

# كويز - 3

1. ما هو الهدف من أنظمة إدارة النسخ(Version Control Systems) ؟
2. اذكر أمثلة على خدمات الحوسبة السحابية.
3. ما هي أنواع المخاطر التي تواجه المشاريع البرمجية؟
4. ما المقصود بالبرمجيات المستدامة(Sustainable Software) ؟

# الأجوبة - 3

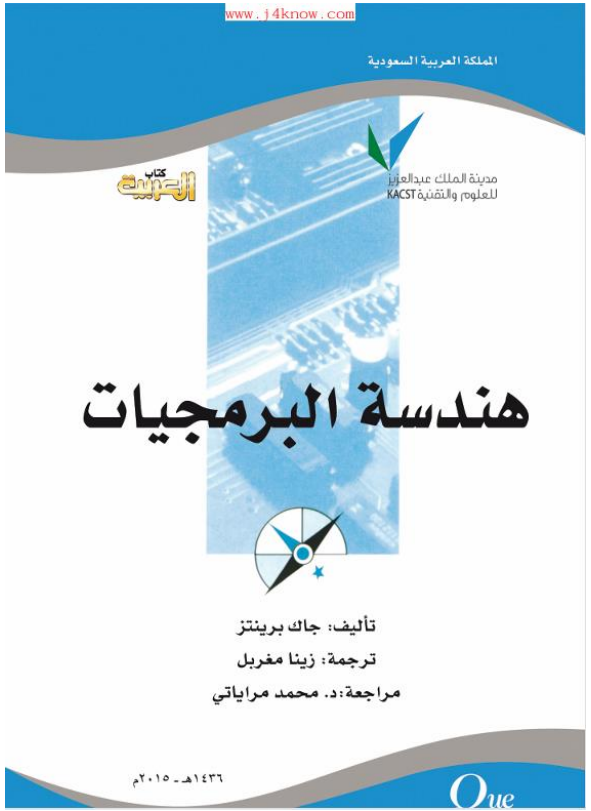
1. تتبع التغييرات في الشيفرة البرمجية واستعادة الإصدارات السابقة.
2. Amazon Web Services (AWS) ,Google Drive
3. مخاطر تقنية: مثل استخدام تقنيات جديدة, مخاطر زمنية: تأخر التسليم, مخاطر مالية: تجاوز الميزانية.
4. برمجيات تقلل استهلاك الطاقة والموارد.

# روابط خارجية

| عنوان الفيديو | الرابط |
| --- | --- |
| هندسة البرمجيات: الفصل الاول: الجزء الاول - مقدمة | https://youtu.be/Db5JWHwetdE?si=_gZDdsOnCTzVWSLe |
| | |
| | |
| | |

# المراجع

- كتاب هندسة البرمجيات من تأليف جاك برينتز



# شكرا لكم