

الوحدة التدريبية الثالثة
المساهمة في عملية تخطيط وتصميم التطبيق
رمز الوحدة
ADS-U3-CAVT

المتطلبات السابقة:

لا يوجد

نتائج التعلم:

عند الانتهاء من دراسة هذه الوحدة واكتساب مهاراتها الأدائية والاتجاهات السلوكية الصحيحة والعلوم المهنية المرافقة من خلال التفاعل مع أنشطتها وخبراتها المختلفة يصبح المتدرب قادراً على أداء نتائج التعلم الآتية:

1. المساهمة في تخطيط مراحل تطوير التطبيقات (جمع المعلومات، التصميم، التطوير/الترميز، الاختبار، النشر، الصيانة)
2. المشاركة في جمع متطلبات العميل حول مدخلات التطبيق والمخرجات والعمليات والمستخدمين ومستوى الوصول / التقييد الخاص بهم
3. المساهمة في اختيار بيئة التطوير المناسبة حسب متطلبات العميل
4. نشر وصيانة أنظمة التطبيق (اتاحة التطبيق للمستخدم والحفاظ على التشغيل مع مرور الزمن)

مصادر التعلم	الأنشطة التعليمية والتدريبية
الوحدة التدريبية	قراءة المعلومات النظرية
الشبكة العنكبوتية	البحث في المواقع الالكترونية التعليمية
منصة الكلية التعليمية	روابط التعليم الالكتروني
المشغل / المختبر	تنفيذ التمارين العملية
سوق العمل	التدريب الميداني

روابط التعلم الإلكتروني

سيتم تزويد المتدربين بروابط التعلم الإلكتروني من خلال المدرب

■ ما الذي يدخل في التخطيط لكل مرحلة من مراحل تطوير التطبيقات ؟

- المساهمة في تخطيط مراحل تطوير التطبيقات (جمع المعلومات، التصميم، التطوير/الترميز، الاختبار، النشر، الصيانة)

1- جمع المعلومات:

يعد جمع المعلومات مرحلة حاسمة في تطوير البرمجيات، حيث تعد الأساس لفهم متطلبات المشروع وتوجيه المراحل اللاحقة من دورة حياة التطوير. سنناقش معاً التقنيات والمنهجيات المختلفة المستخدمة لجمع المعلومات الأساسية من أصحاب المصلحة والمستخدمين النهائيين والمصادر الأخرى ذات الصلة.

- تحديد أصحاب المصلحة:

- أ. تعريف أصحاب المصلحة: تحديد الأفراد أو المجموعات ذات المصالح الخاصة في مشروع البرنامج.
- ب. تحليل احتياجات أصحاب المصلحة: تقنيات تحليل احتياجات أصحاب المصلحة وتوقعاتهم وتأثيرهم على المشروع.
- ج. استراتيجيات إشراك أصحاب المصلحة: إنشاء قنوات اتصال والحفاظ على المشاركة النشطة طوال عملية التطوير.

- متطلبات الاستنباط:

- أ. فهم احتياجات المستخدم: إجراء المقابلات والاستبيانات والملاحظات لتحديد متطلبات المستخدم وتفضيلاته.
- ب. جمع المتطلبات الوظيفية: توثيق الميزات والوظائف وسلوكيات النظام التي يرغب فيها المستخدمون.
- ج. المتطلبات غير الوظيفية: التقاط القيود مثل معايير الأداء والأمان وسهولة الاستخدام.

- تقنيات التوثيق:

- أ. المستندات المطلوبة: إنشاء مستندات منظمة لالتقاط وتنظيم المعلومات المجمعة، بما في ذلك قصص المستخدم وحالات الاستخدام والشخصيات.
- ب. النماذج الأولية: تطوير نماذج بالحجم الطبيعي أو إطارات سلوكية أو نماذج أولية لتصوير وظائف البرنامج وجمع التعليقات في وقت مبكر من العملية.
- ج. القصة المصورة: إنشاء قصص مرئية لتوضيح تفاعلات المستخدم وسير عمل النظام.

- طرق جمع البيانات:

- أ. المسوحات: تصميم وتوزيع المسوحات لجمع البيانات الكمية والنوعية من جمهور كبير.

- ب. المقابلات: إجراء مقابلات فردية أو جماعية مع أصحاب المصلحة للتعلم في متطلبات وتفضيلات محددة.
- ج. ورش العمل ومجموعات التركيز: تسهيل الجلسات التعاونية لتبادل الأفكار، وتحديد أولويات المتطلبات، وحل وجهات النظر المتضاربة.
- د. الملاحظات: مراقبة المستخدمين في بيئتهم الطبيعية لفهم سير العمل ونقاط الضعف وأنماط الاستخدام.

• الأدوات والتقنيات:

- أ. أدوات إدارة المتطلبات: تقديم أدوات برمجية مثل JIRA أو Trello أو Microsoft Azure DevOps لتوثيق متطلبات المشروع وتتبعها وإدارتها.
- ب. منصات الاستطلاع وجمع البيانات: استكشاف أدوات مثل Google Forms أو SurveyMonkey أو Qualtrics لإنشاء الاستطلاعات وتحليلها.
- ج. منصات الاتصال: استخدام أدوات الاتصال مثل Slack أو Microsoft Teams أو Zoom لتسهيل اجتماعات ومناقشات أصحاب المصلحة.

• التحقق من المتطلبات وتتبعها:

- أ. التحقق من صحة المتطلبات: التأكد من أن المتطلبات المجمعة تعكس بدقة احتياجات أصحاب المصلحة وأنها ممكنة ضمن قيود المشروع.
- ب. تحديد أولويات المتطلبات: تحديد أولويات المتطلبات بشكل تعاوني بناءً على أهميتها وإلحاحها وتأثيرها على نجاح المشروع.
- ج. إمكانية تتبع المتطلبات: إنشاء إمكانية التتبع بين المتطلبات وعناصر التصميم والتطوير والاختبار المقابلة لضمان التوافق طوال دورة حياة المشروع.

2- التصميم في مراحل تطوير التطبيقات:

يعد التصميم مرحلة حاسمة في تطوير البرمجيات حيث يتم تحويل المفاهيم والمتطلبات إلى مخطط لبناء نظام البرمجيات. يركز هذا الموضوع المبادئ والمنهجيات وأفضل الممارسات المتبعة في تصميم الحلول البرمجية التي تلبي احتياجات المستخدم وأهداف المشروع. وهناك عدة أنواع من التصميم منها:

• التصميم المعماري:

- أ. التعريف: نظرة عامة على التصميم المعماري مع التركيز على البنية والتنظيم رفيعي المستوى لنظام البرمجيات.
- ب. الأنماط المعمارية: استكشاف الأنماط المعمارية الشائعة مثل خادم العميل، والهندسة المعمارية ذات الطبقات، والخدمات الصغيرة، والهندسة المعمارية القائمة على الأحداث.

ج. مبادئ التصميم: تطبيق مبادئ مثل النمطية والتغليف والفصل بين الاهتمامات لضمان بنية قابلة للتطوير وقابلة للصيانة.

د. المفاضلات المعمارية: تقييم المفاضلات بين الأنماط والأنماط المعمارية المختلفة بناءً على متطلبات المشروع وقابلية التوسع والأداء وقابلية الصيانة.

● التصميم التفصيلي:

- أ. تصميم المكونات: تقسيم النظام إلى مكونات/وحدات أصغر وتحديد واجهاتها ومسؤولياتها وتفاعلاتها.
- ب. أنماط التصميم: تقديم أنماط التصميم مثل الأنماط الإبداعية والهيكلية والسلوكية لمعالجة مشاكل التصميم المتكررة وتعزيز إعادة استخدام التعليمات البرمجية.
- ج. تصميم الفئات والأشياء: تصميم الفئات والأشياء وعلاقاتها باستخدام مبادئ مثل الميراث (inheritance) وتعدد الأشكال (polymorphism).
- د. لغة النمذجة الموحدة (UML) Unified Modeling Language : إنشاء مخططات لغة النمذجة الموحدة (UML) مثل مخططات الفئات، ومخططات التسلسل، ومخططات الأنشطة لتصوير قرارات التصميم وتوصيلها.

● تصميم واجهة المستخدم (UI) User Interface:

- أ. التصميم الذي يركز على المستخدم (UCD): فهم احتياجات المستخدم وتفضيلاته وسلوكياته لتصميم واجهات بديهية وسهلة الاستخدام.
- ب. التخطيط السلوكي والنماذج الأولية: إنشاء إطارات سلوكية ونماذج أولية تفاعلية لتصوير تخطيطات واجهة المستخدم وتدفقات التنقل وأنماط التفاعل.
- ج. مبادئ تصميم واجهة المستخدم: تطبيق مبادئ سهلة الاستخدام، وإمكانية الوصول والجماليات لإنشاء واجهات مستخدم جذابة وفعالة.
- د. التصميم سريع الاستجابة: تصميم واجهات تتكيف مع أحجام الشاشات والأجهزة المختلفة لضمان تجربة مستخدم متسقة عبر الأنظمة الأساسية.

● تصميم قاعدة البيانات:

- أ. نمذجة العلاقة بين الكيان (Entity-relationship modeling (ERM) : نمذجة بنية البيانات والعلاقات باستخدام تقنيات مثل مخططات العلاقة بين الكيانات (ERDs).
- ب. التطبيق: تطبيق تقنيات التطبيق لتقليل تكرار البيانات وضمان سلامة البيانات في مخططات قواعد البيانات العلائقية.
- ج. الفهرسة والتحسين: تصميم مخططات قاعدة بيانات فعالة عن طريق تحسين أداء الاستعلام، وفهرسة الأعمدة الرئيسية، وإلغاء التسوية عند الضرورة.

د. إدارة المعاملات: تصميم أنظمة المعاملات للحفاظ على اتساق البيانات وسلامتها من خلال تقنيات مثل خصائص ACID والتحكم في التزامن.

• التصميم الأمني:

- أ. نمذجة التهديدات: تحديد التهديدات الأمنية المحتملة ونقاط الضعف في نظام البرمجيات من خلال تمارين نمذجة التهديدات.
- ب. الضوابط الأمنية: تنفيذ الضوابط الأمنية مثل المصادقة والترخيص والتشفير والتحقق من صحة المدخلات للتخفيف من المخاطر المحددة.
- ج. ممارسات الترميز الآمن: الالتزام بإرشادات الترميز الآمن وأفضل الممارسات لمنع الثغرات الأمنية الشائعة مثل حقن SQL، والبرمجة النصية عبر المواقع (XSS)، وتجاوز سعة المخزن المؤقت.
- د. متطلبات الامتثال: التأكد من أن تصميم البرنامج يلي معايير الأمان التنظيمية والخاصة بالصناعة مثل اللائحة العامة لحماية البيانات (GDPR) وقانون HIPAA و PCI DSS.

• توثيق التصميم:

- أ. مستندات التصميم: إنشاء وثائق تصميم شاملة بما في ذلك المخططات المعمارية ومواصفات المكونات ونماذج واجهة المستخدم ومخططات قاعدة البيانات.
- ب. مراجعات التصميم: إجراء مراجعات التصميم مع أصحاب المصلحة وأعضاء فريق التطوير للحصول على التعليقات وتحديد المشكلات وتحسين التصميم.
- ج. التحكم في الإصدار: إدارة عناصر التصميم والمراجعات باستخدام أنظمة التحكم في الإصدار مثل Git لتتبع التغييرات والتعاون بشكل فعال.

3- التطوير/الترميز Coding :

التطوير، المعروف أيضًا باسم البرمجة، هو مرحلة في هندسة البرمجيات حيث تتم ترجمة التصميمات والمتطلبات إلى تطبيقات برمجية فعلية. يركز هذا الموضوع على المبادئ والمنهجيات وأفضل الممارسات المتبعة في كتابة تعليمات برمجية عالية الجودة تكون فعالة وقابلة للصيانة وقابلة للتطوير.

• لغات البرمجة:

- أ. نظرة عامة على لغات البرمجة شائعة الاستخدام في تطوير البرامج، بما في ذلك اللغات عالية المستوى مثل Java و Python و C# و JavaScript، بالإضافة إلى اللغات الخاصة بالمجال (DSL) ولغات البرمجة النصية.
- ب. معايير الاختيار: العوامل التي يجب مراعاتها عند اختيار لغة البرمجة، مثل متطلبات المشروع، وخبرة الفريق، واعتبارات الأداء، ودعم النظام البيئي.

• معايير واتفاقيات الترميز:

- أ. أهمية معايير الترميز: ضمان الاتساق وسهولة القراءة وقابلية الصيانة للتعليمات البرمجية عبر فريق التطوير.
- ب. الاتفاقيات المعتمدة: وضع مبادئ توجيهية لاصطلاحات التسمية، وتنسيق التعليمات البرمجية، والمسافات البادئة، والتعليق، والتوثيق.
- ج. أدوات فرض المعايير: استخدام أدوات فحص التعليمات البرمجية وملحقات IDE وعمليات مراجعة التعليمات البرمجية الآلية لفرض معايير الترميز واكتشاف الأخطاء الشائعة.

• التحكم في الإصدار:

- أ. أنظمة التحكم في الإصدار (VCS): مقدمة إلى Git و Subversion (SVN) وأدوات VCS الأخرى لإدارة تغييرات التعليمات البرمجية والتعاون وسجل الإصدارات.
- ب. إستراتيجيات التفرع: فهم إستراتيجيات التفرع والدمج مثل تفرع الميزات، وتفرع الإصدار، و Gitflow لتسهيل التطوير الموازي وإدارة التعليمات البرمجية.
- ج. سير العمل التعاوني: تنفيذ سير العمل مثل طلبات السحب ومراجعات التعليمات البرمجية والتكامل المستمر (CI) لتعزيز التعاون وضمان جودة التعليمات البرمجية.

• منهجيات تطوير البرمجيات:

- أ. المنهجيات الرشيقة: Agile نظرة عامة على ممارسات Agile مثل Scrum و Kanban و Extreme Programming (XP) التي تركز على التطوير التكراري وتعاون العملاء والقدرة على التكيف مع المتطلبات المتغيرة.
- ب. منهجيات الشلال مقابل المنهجيات الرشيقة: مقارنة تطوير Waterfall التقليدي مع منهجيات Agile من حيث إدارة المشروع، وتخفيف المخاطر، ورضا العملاء.
- ج. الأساليب الهجينة: استكشاف الأساليب المختلطة التي تجمع بين عناصر المنهجيات الرشيقة (Agile) والتقليدية لتناسب سياقات وقيود المشروع المحددة.

• التطوير القائم على الاختبار (TDD) واختبار الوحدة:

- أ. التطوير المبني على الاختبار (TDD): فهم دورة TDD لكتابة الاختبارات، وكتابة التعليمات البرمجية لاجتياز الاختبارات، وإعادة البناء لتحسين التصميم وقابلية الصيانة.
- ب. أطر اختبار الوحدة: مقدمة لأطر اختبار الوحدة مثل JUnit و NUnit و pytest لكتابة الاختبارات وأتمتها على مستوى الوحدة.
- ج. تغطية الاختبار: قياس تغطية التعليمات البرمجية للتأكد من اختبار مسارات التعليمات البرمجية الهامة وتحديد المناطق لتغطية الاختبار الإضافية.

• جودة الكود وإعادة البناء:

- أ. ممارسات مراجعة الكود: إجراء مراجعات الكود لتحديد الأخطاء والمشكلات المعمارية والالتزام بمعايير الترميز وتعزيز تبادل المعرفة والتحسين المستمر.
- ب. تقنيات إعادة البناء: إعادة بناء التعليمات البرمجية لتحسين إمكانية القراءة وقابلية الصيانة والأداء دون تغيير سلوكها الخارجي، مسترشدة بمبادئ مثل SOLID و DRY
- ج. أدوات تحليل التعليمات البرمجية: استخدام أدوات تحليل التعليمات البرمجية الثابتة مثل SonarQube و ESLint و ReSharper لتحديد التعليمات البرمجية والأخطاء المحتملة وفرص التحسين.

4-الاختبارات:

يعد الاختبار مرحلة حاسمة في هندسة البرمجيات يهدف إلى تحديد العيوب والأخطاء في نظام البرنامج لضمان جودته وموثوقيته وأدائه. يركز هذا الموضوع على المبادئ والمنهجيات وأفضل الممارسات المتبعة في اختبار البرامج لتقديم حلول برمجية عالية الجودة.

• أساسيات الاختبار:

- أ. أهمية الاختبار: فهم أهمية الاختبار في ضمان جودة البرمجيات ورضا العملاء ونجاح الأعمال.
- ب. أهداف الاختبار: تحديد الأهداف الأساسية للاختبار، بما في ذلك اكتشاف العيوب، والتحقق من صحة المتطلبات، والتحقق من الأداء الوظيفي.
- ج. مبادئ الاختبار: استكشاف مبادئ الاختبار الأساسية مثل الاختبار المبكر، والاختبار الشامل، وتجميع العيوب.

• أنواع الاختبار:

- أ. اختبار الوحدة: اختبار المكونات الفردية أو وحدات التعليمات البرمجية بشكل منفصل للتحقق من صحتها ووظائفها.
- ب. اختبار التكامل: التحقق من التفاعلات بين الوحدات أو المكونات المختلفة للتأكد من أنها تعمل معًا كما هو متوقع.
- ج. اختبار النظام: اختبار نظام البرنامج بأكمله للتحقق من امتثاله للمتطلبات والسلوك المحدد.
- د. اختبار القبول: تقييم جاهزية البرنامج للنشر والقبول من قبل المستخدمين النهائيين أو أصحاب المصلحة.
- هـ. اختبار الأداء: تقييم أداء البرنامج وقابلية التوسع والموثوقية في ظل ظروف التحميل والضغط المختلفة.
- و. اختبار الأمان: تحديد نقاط الضعف في آليات أمان البرنامج لمنع الوصول غير المصرح به وانتهاك البيانات.

• استراتيجيات الاختبار:

- أ. اختبار الصندوق الأسود: اختبار وظائف البرنامج دون معرفة التنفيذ الداخلي له، مع التركيز على المدخلات والمخرجات والسلوك.
- ب. اختبار الصندوق الأبيض: فحص البنية الداخلية ومنطق البرنامج للتأكد من ممارسة جميع مسارات التعليمات البرمجية والتحقق من صحتها.
- ج. اختبار الصندوق الرمادي: الجمع بين عناصر اختبار الصندوق الأسود والصندوق الأبيض لتحقيق تغطية اختبار شاملة مع الاستفادة من المعرفة بالأعمال الداخلية للنظام.

• تخطيط وإدارة الاختبار:

- أ. تخطيط الاختبار: وضع خطة اختبار تحدد أهداف الاختبار ونطاقه وموارده وجدوله الزمني والتسليمات.
- ب. تصميم حالة الاختبار: إنشاء حالات اختبار تفصيلية تحدد مدخلات الاختبار والنتائج المتوقعة وشروط الاختبار لكل سيناريو اختبار.
- ج. تنفيذ الاختبار: تنفيذ حالات الاختبار وتسجيل نتائج الاختبار وتوثيق أية عيوب أو انحرافات عن السلوك المتوقع.
- د. تقارير الاختبار: إنشاء تقارير اختبار تلخص تغطية الاختبار وحالة النجاح/الفشل ومقاييس العيوب والتوصيات لمزيد من التحسين.

• أتمتة الاختبار:

- أ. أطر عمل أتمتة الاختبار: مقدمة إلى أطر عمل أتمتة الاختبار مثل Selenium وJUnit وpytest لأتمتة حالات الاختبار المتكررة واختبار الانحدار.
- ب. فوائد أتمتة الاختبار: استكشاف مزايا أتمتة الاختبار، بما في ذلك زيادة تغطية الاختبار، ودورات ردود الفعل الأسرع، وتقليل الجهد اليدوي.
- ج. اختبار البرمجة النصية: كتابة وصيانة نصوص اختبارية آلية باستخدام لغات البرمجة وأطر الاختبار للتحقق من صحة وظائف البرنامج وسلوكه.

• الاختبار المستمر وDevOps:

- أ. التكامل المستمر (CI): دمج الاختبار الآلي في عملية تطوير البرامج لاكتشاف العيوب وإصلاحها في وقت مبكر من دورة التطوير.
- ب. النشر المستمر (CD): أتمتة نشر وإصدار تحديثات البرامج إلى بيئات الإنتاج، مصحوبًا باختبار شامل لضمان الاستقرار والموثوقية.
- ج. ممارسات DevOps: احتضان مبادئ DevOps مثل التعاون والأتمتة والتحسين المستمر لتبسيط دورة حياة تطوير البرامج وتقديم القيمة للعملاء بشكل أكثر كفاءة.

5-نشر تطبيقات البرمجيات:

النشر هو مرحلة في هندسة البرمجيات حيث يتم إصدار البرنامج المطور وإتاحته للاستخدام من قبل المستخدمين النهائيين. يركز هذا الموضوع على المبادئ والمنهجيات وأفضل الممارسات المتبعة في نشر التطبيقات البرمجية بفعالية وكفاءة.

• تخطيط النشر:

- أ. تعريف النشر: فهم عملية النشر وأهميتها في تقديم الحلول البرمجية للمستخدمين النهائيين.
- ب. استراتيجية النشر: تحديد استراتيجية النشر الأكثر ملاءمة بناءً على متطلبات المشروع وقدرات البنية التحتية وبيئة النشر.
- ج. تكوين البيئة: إعداد بيئات التطوير والاختبار والتدريب والإنتاج لتسهيل النشر وضمان الاتساق عبر المراحل المختلفة لخط أنابيب النشر.

• أتمتة البناء:

- أ. التكامل المستمر (CI): دمج عمليات البناء الآلية في سير عمل التطوير لتجميع تغييرات البرامج وتعبئتها واختبارها تلقائيًا.
- ب. أدوات البناء: استخدام أدوات أتمتة البناء مثل Jenkins أو TeamCity أو Travis CI لأتمتة مهام البناء وتبسيط مسار النشر.
- ج. إدارة القطع الأثرية: إدارة عناصر البناء والتبعيات باستخدام مستودعات القطع الأثرية مثل Nexus أو Artifactory لضمان التحكم في الإصدار وإمكانية التتبع.

• الحاويات والتنسيق: الحاويات هي مكونات برمجية خفيفة الوزن تعمل بكفاءة

- أ. النقل بالحاويات: تعبئة التطبيقات البرمجية وتبعياتها في حاويات خفيفة الوزن ومحمولة باستخدام تقنيات مثل Docker أو Podman.
- ب. تنسيق الحاويات: إدارة التطبيقات الموجودة في حاويات وتوسيع نطاقها عبر مضيفين متعددين باستخدام منصات تنسيق الحاويات مثل Kubernetes أو Docker Swarm.
- ج. فوائد النقل بالحاويات: استكشاف مزايا النقل بالحاويات، بما في ذلك الاتساق والعزل وقابلية التوسع وكفاءة الموارد.

• استراتيجيات النشر:

- أ. النشر الأزرق والأخضر (Blue-Green Deployment): طرح الإصدارات الجديدة عن طريق تحويل حركة المرور بين بيئتين متطابقتين (الأزرق والأخضر) لتقليل وقت التوقف عن العمل والمخاطر.
- ب. نشر Canary: إطلاق ميزات أو تحديثات جديدة تدريجيًا لمجموعة فرعية من المستخدمين أو الخوادم لجمع الملاحظات والتخفيف من المشكلات المحتملة قبل النشر الكامل.
- ج. النشر المستمر: نشر إصدارات جديدة من البرامج تدريجيًا عبر خوادم أو مثيلات متعددة لتقليل التأثير على توفر النظام وأدائه.
- د. اختبار أ/ب (A/B Testing): تجربة إصدارات مختلفة من البرامج في وقت واحد لمقارنة أدائها وتجربة المستخدم وفعاليتها.

• المراقبة والتسجيل:

- أ. أدوات المراقبة: تنفيذ أدوات المراقبة مثل Prometheus أو Grafana أو Nagios لتتبع صحة النظام ومقاييس الأداء والتوفر.
- ب. إدارة السجل: جمع وتجميع وتحليل السجلات التي تم إنشاؤها بواسطة التطبيقات المنشورة باستخدام أطر عمل التسجيل مثل ELK (Elasticsearch أو Logstash أو Kibana) أو Splunk.
- ج. التنبيهات والإشعارات: تكوين آليات التنبيه لإعلام المسؤولين أو أصحاب المصلحة بالأحداث الهامة أو الحالات الشاذة أو تدهور الأداء في البيئة المنشورة.

• التراجع والتعافي من الكوارث:

- أ. إجراءات التراجع: وضع إجراءات وآليات التراجع للعودة إلى إصدارات البرامج أو التكوينات السابقة في حالة فشل النشر أو حدوث مشكلات غير متوقعة.
- ب. التخطيط للتعافي من الكوارث: تطوير خطط واستراتيجيات التعافي من الكوارث للتعافي من الأحداث الكارثية أو فقدان البيانات أو فشل البنية التحتية التي تؤثر على البيئة المنشورة.
- ج. النسخ الاحتياطي والاستعادة: تنفيذ نسخ احتياطية منتظمة للبيانات والتكوينات والمكونات المهمة لتسهيل الاسترداد السريع وتقليل وقت التوقف عن العمل أثناء سيناريوهات الكوارث.

6-صيانة:

تعد الصيانة مرحلة حاسمة في هندسة البرمجيات والتي تتضمن إدارة التطبيقات البرمجية وتحديثها وتحسينها لضمان استمرار وظائفها وموثوقيتها وأهميتها بمرور الوقت. يستكشف هذا الموضوع المبادئ والمنهجيات وأفضل الممارسات المتبعة في صيانة البرامج للحفاظ على أنظمة البرامج وتحسينها طوال دورة حياتها.

• أنواع الصيانة:

- أ. الصيانة التصحيحية: معالجة وإصلاح العيوب أو الأخطاء التي تم تحديدها أثناء التشغيل لاستعادة البرنامج إلى حالة العمل.
- ب. الصيانة التكيفية: تعديل البرنامج لاستيعاب التغييرات في بيئة التشغيل، مثل ترقيات الأجهزة، أو تحديثات نظام التشغيل، أو المتطلبات التنظيمية.
- ج. الصيانة المثالية: تعزيز وتحسين وظائف البرنامج أو أدائه أو سهولة استخدامه بناءً على تعليقات المستخدمين أو المتطلبات المتطورة أو التقدم التكنولوجي.
- د. الصيانة الوقائية: تحديد المشكلات أو المخاطر المحتملة بشكل استباقي والتخفيف من حدتها من خلال إجراءات مثل إعادة هيكلة التعليمات البرمجية وتحسين الأداء والتحديثات الأمنية.

• إدارة التغيير:

- أ. إدارة طلبات التغيير: تنفيذ عملية رسم اليد لتقديم طلبات التغيير وتقييمها وتحديد أولوياتها والموافقة عليها لضمان صيانة البرامج بشكل منظم ومنضبط.
- ب. التحكم في الإصدار: إدارة التغييرات على عناصر البرامج والتكوينات والوثائق باستخدام أنظمة التحكم في الإصدار مثل Git لتتبع المراجعات وإدارة التعارضات وضمان إمكانية التتبع.
- ج. تحليل التأثير: تقييم التأثير المحتمل للتغييرات المقترحة على وظائف البرنامج وأدائه وموثوقيته وأمانه لإرشاد عملية صنع القرار وإدارة المخاطر.

• تتبع الأخطاء وحلها:

- أ. الإبلاغ عن الأخطاء: وضع إجراءات للمستخدمين وأصحاب المصلحة للإبلاغ عن الأخطاء أو العيوب أو المشكلات التي تمت مواجهتها أثناء تشغيل البرنامج من خلال أنظمة مخصصة لتتبع الأخطاء أو بوابات مكتب المساعدة.
- ب. فرز الأخطاء: تحديد أولويات الأخطاء المبلغ عنها وتصنيفها بناءً على مدى خطورتها وتأثيرها ومدى إلحاحها لتخصيص الموارد بكفاءة ومعالجة المشكلات المهمة على الفور.
- ج. حل الأخطاء: التحقيق في الأخطاء المبلغ عنها وتشخيصها وإصلاحها باستخدام أدوات تصحيح الأخطاء وتقنيات تحليل التعليمات البرمجية والتعاون مع فرق التطوير لضمان الحل في الوقت المناسب وضمان الجودة.

• تحديثات البرامج وترقياتها:

- أ. إدارة التصحيح: نشر التصحيحات والإصلاحات العاجلة والتحديثات الأمنية لمعالجة نقاط الضعف المعروفة، وتخفيف المخاطر الأمنية، وضمان استمرار سلامة البرنامج وأمانه.

ب. ترقية الإصدار: تخطيط وتنفيذ ترقية الإصدار أو الترحيل إلى الإصدارات الأحدث من أطر البرامج أو المكتبات أو الأنظمة الأساسية للاستفادة من الميزات الجديدة وتحسينات الأداء وإصلاحات الأخطاء.

ج. اختبار التوافق: التحقق من صحة تحديثات البرامج أو ترقيةها عبر بيئات وتكوينات وتبعيات مختلفة لضمان التوافق مع الأنظمة الحالية ومسارات العمل.

● مراقبة الأداء وتحسينه:

أ. مراقبة الأداء: تنفيذ أدوات ومقاييس المراقبة لتتبع أداء البرامج واستخدام الموارد وأوقات الاستجابة والإنتاجية في الوقت الفعلي لتحديد اختناقات الأداء وفرص التحسين.

ب. ضبط الأداء: تحليل بيانات الأداء، وكود التوصيف، وتطبيق تقنيات التحسين مثل التخزين المؤقت، وفهرسة قاعدة البيانات، وإعادة هيكلة التعليمات البرمجية، وتحسين الخوارزمية لتحسين أداء البرنامج وقابلية التوسع.

ج. تخطيط القدرات: التنبؤ بمتطلبات الموارد المستقبلية وقدرات التوسع بناءً على البيانات التاريخية وأنماط الاستخدام وتوقعات النمو والتغيرات المتوقعة في طلب المستخدم أو عبء عمل النظام.

● التخطيط لنهاية عمر التطبيق وإيقافه:

أ. تقييم نهاية العمر: تقييم جدوى المنتج البرمجي واستدامته وأهميته في ضوء ديناميكيات السوق المتغيرة والتقدم التكنولوجي وأولويات العمل لتحديد مساره المستقبلي.



ب. التخطيط للتقاعد: تطوير نهج تدريجي لإيقاف أنظمة البرامج القديمة، بما في ذلك ترحيل البيانات، وانتقال المستخدم، ونقل المعرفة، وأنشطة إيقاف التشغيل لتقليل التعطيل وتحقيق أقصى قدر من القيمة.

ج. الدعم القديم: توفير الدعم المستمر والصيانة والتحديثات الأمنية لأنظمة البرامج القديمة بعد انتهاء عمرها الافتراضي لتلبية احتياجات العمل الهامة أو متطلبات الامتثال أو الالتزامات التعاقدية.

• تمرين عملي في نهاية الوحدة

اسم التمرين: إنشاء مخطط تطبيق	رقم التمرين: 1/3
الزمن المخصص لتنفيذ التمرين:	
اهداف التمرين العملي:	
<ul style="list-style-type: none"> • تصميم واجهة سهلة الاستخدام لإدخال التمارين وجدولتها • اختيار التقنيات والادوات • تتبع التقدم اليومي أو الأسبوعي • تحليل البيانات وعرض الإحصائيات. • عرض رسوم بيانية توضح تقدم المستخدم نحو أهدافه 	
تعليمات المدرب:	
<p>يقدم المدرب توضيح للخطوات حسب تعليمات السلامة المتبعة في المملكة الاردنية الهاشمية ويراعي ان يوضح كل من التعليمات التالية والتي تتفق مع اجراءات السلامة حسب القوانين المرعية.</p> <ul style="list-style-type: none"> • الخروج من موقع العمل عند سماع جرس الانذار. • اسلك الطرف الايسر من الممرات الامنة و اترك الطرف الايمن خاليا • عند الخروج من الغرف تفقد اتجاه الخطر قبل الخروج مثلا تفقد يد الباب عند الحريق للتأكد من عدم وجود حرارة مرتفعة. • اغلاق الباب عند الخروج من الاماكن المغلقة لمنع انتشار الحريق • الصراخ عند الخروج او استخدام مكبر الصوت منها الاخرين بكلمة اخلاء مع التكرير عند كل نقطة • الاتصال بالدفاع المدني اذا تطلبت الحاجة يوضع رقم الدفاع المدني امام المتدربين بطريقة ظاهرة , رقم الطوارئ الموحد 911. • التوجه الى مواقع التجمع المحددة. • تفقد فريق عملك وحدد الغائبين. • ابلاغ فريق الانقاذ بأعداد الحضور واسماء الغائبين. 	
التسهيلات التدريبية اللازمة لتنفيذ التمرين العملي :	

المواد الأولية	العدد والادوات	الاجهزة والآلات
<ul style="list-style-type: none"> • قرطاسية (ورق للكتابة • ، اقلام حبر، اقلام تخطيط • أدوات العصف الذهني مثل السبورة الورقية Flip Chart 	<p>أجهزة الحاسوب، الشاشات، انترنت</p> <p>عالي السرعة، سماعات الرأس، كاميرات الويب، الواح تفاعلية، طابعات، ماسحات ضوئية</p>	<p>✓ أدوات الاتصال والمراسلة</p> <ul style="list-style-type: none"> • أدوات ادارة المشروع • أدوات التحرير والتوثيق مثل Microsoft Office • منصات التواصل الاجتماعي <p>أدوات لتحليل أنماط التعاون ومقاييس الاتصال وتقدم المشروع مثل , Google Analytics Microsoft Power point</p>

خطوات العمل لتنفيذ التمرين	
الشكل / الصورة	خطوات العمل ومعايير ادائها (الرقمية & الوصفية)
	تصميم واجهة المستخدم وإنشاء النماذج الأولية.
	تطوير الميزات الأساسية مثل إنشاء خطط التمارين وتتبع التقدم

	<p>اختبار التطبيق وإصلاح الأخطاء</p>
	<p>إطلاق التطبيق على متاجر التطبيقات</p>
<p>تقييم جودة المنتج: وهي مواصفات جودة ودقة ناتج الأداء المطلوب سواء كان منتج أو خدمة</p>	
	<p>وضوح الأهداف والتخطيط</p>
	<p>تفاعل المشاركين والمشاركة الفعالة</p>
	<p>جودة المحتوى والأدوات المستخدمة</p>
	<p>تأثير التمرين على المشاركين</p>
	<p>التقييم والتغذية الراجعة</p>
<p>تقييم الاتجاهات والسلامة المهنية: وهي الاتجاهات المتعلقة بالمحافظة على الأجهزة والأدوات وترتيبها وتنظيفها وتخزينها والسلامة المهنية المتعلقة بالأشخاص والآلات والمعدات والمواد أثناء تأدية التمرين العملي</p>	
	<p>الوعي والالتزام بقواعد السلامة</p>
	<p>السلوكيات والممارسات اليومية</p>
	<p>الثقافة العامة للسلامة المهنية</p>
	<p>تقييم بيئة العمل</p>
	<p>الالتزام باللوائح والقوانين</p>
	<p>التدريب والتطوير</p>

تقييم التمرين العملي رقم (1/3) ترفق بعد كل بطاقة تمرين عملي

قيم ادائك بعد انتهائك من تنفيذ التمرين من خلال:

- ضع إشارة " ✓ " بجانب الخطوات التي اتقنتها.
- ضع إشارة " × " بجانب الخطوات التي لم تتقنها مع ذكر الاسباب وكيف يمكنك تحسينها بالمناقشة مع مدربك
- ضع إشارة " × " بجانب الخطوات غير القابلة للتطبيق مع ذكر الاسباب المؤدية لعدم قابلية تطبيق وتنفيذ الخطوة وكيف يمكنك اعادة تطبيقها باتقان وتحسينها بالمناقشة مع مدربك.

الرقم	اسم الخطوة (عليك اتقان جميع الخطوات أدناه حتى تظهر درجة الإتقان للأداء المطلوب في التمرين)	لا اتقنها	غير قابلة للتطبيق	ذكر اسباب عدم الاتقان ولماذا هي غير قابلة للتطبيق
1	هل تم تحديد هدف التمرين			
2	هل استطاع شرح أهمية إنشاء تطبيق مخطط			
3	هل قام بتحليل استخدام الوقت الحالي			
4	هل قام بتقديم أدوات وتقنيات إنشاء تطبيق مخطط			
5	هل تم توضيح الأهداف والتخطيط			
6	هل كان جودة المحتوى والأدوات المستخدمة جيدة			
7	هل هنالك التزام بقواعد السلامة			
8	السلوكيات والممارسات اليومية			
9	الثقافة العامة للسلامة المهنية			
10	هل بيئة العمل جيدة			
11	هل تم الالتزام باللوائح والقوانين			
<p>أنجزت اداء التمرين العملي المطلوب</p>				
	ضمن الوقت المحدد	زيادة عن الوقت المحدد	غير قابل للإنجاز	ذكر اسباب: ✓ زيادة عن الوقت المحدد ✓ ولماذا التمرين غير قابل للإنجاز