Intro


apptrainers
Translating ideas to reality through innovative education

Business modeling

Business analysis

1. **Requirements Gathering (جمع المتطلبات)**
   - يجتمع محلل الأعمال (BA) مع العميل وأصحاب المصلحة (stakeholders) لفهم المشروع.
   - يتم توثيق جميع المتطلبات في مستند يسمى **SRS (Software Requirement Specification)**.

2. **Planning (التخطيط)**
   التخطيط لبناء المشروع:
   - ما الذي سيتم تطويره أولًا؟
   - ما المهام التي تحتاج إلى إنجاز؟
   - متى يجب الانتهاء من كل جزء؟
   - تُستخدم أدوات مثل **JIRA** أو **Trello** لتنظيم العمل.

3. **Task Management (إدارة المهام)**
   - يتم تقسيم المشروع إلى مهام صغيرة يمكن تنفيذها.
   - كل مهمة مسؤول عنها شخص معين، مثل:
     - المطورين (Developers) لكتابة الكود من فرونت إند و باك إند.
     - المصممين (UI/UX Designers) لتصميم الواجهات.
     - المختبرين (QA) لاختبار الجودة.

4. **Analysis Delivering (تسليم التحليل)**
   - يقدم محلل الأعمال (BA) تقريرًا يوضح:
     - المتطلبات النهائية.
     - خطة التنفيذ.
     - أي مخاطر متوقعة.

   - يتم مناقشة التقرير مع الفريق التقني لضمان فهم الجميع لما سيتم تطويره.

Development teams

الأشخاص المسؤولين عن كتابة **source code** سواء كان **front end** أو **back end developers**

بعد استلام المتطلبات من فريق **Business Analysis (BA)**، يأتي دور فريق التطوير لتحويل الفكرة إلى منتج فعلي. الفريق يتكون من عدة أدوار، وكل دور مسؤول عن جزء معين من المشروع.

1. **Frontend Developers (مطوروا الواجهة الأمامية)**
   - تحويل التصاميم إلى صفحات تفاعلية باستخدام **HTML, CSS, JavaScript**

2. **Backend Developers**
   - بناء الخوادم وقواعد البيانات وإدارة العمليات الخلفية.
   - تطوير **APIs** التي تتفاعل مع الواجهة الأمامية.
   - تحسين الأداء ومعالجة الأمان مثل **Authentication & Authorization**.

3. **Full-Stack Developers**
   - لديهم خبرة في تطوير الواجهة الأمامية والخلفية معًا.
   - يربطون بين مختلف أجزاء التطبيق، ويضمنون تكامل البيانات بين **Frontend** و **Backend**.

4. **Mobile Developers (مطوروا تطبيقات الموبايل)**
   - تطوير تطبيقات موبايل لكل من **iOS** و **Android**

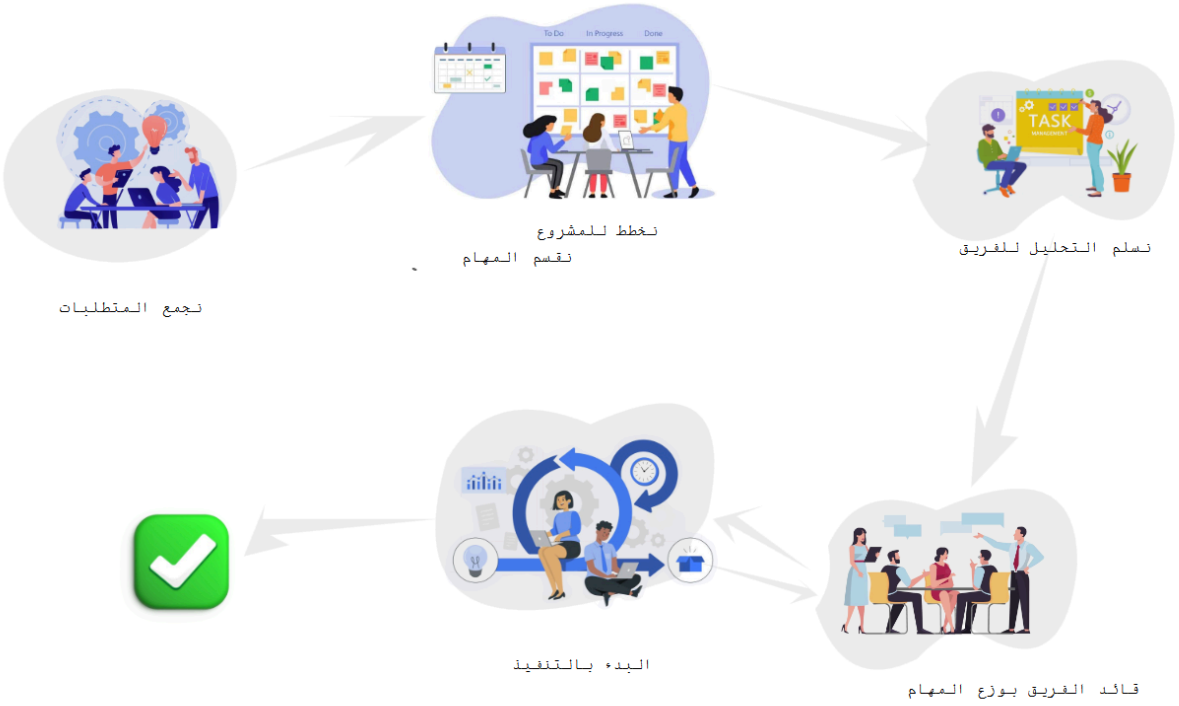
5. **فريق الاختبار (Testing / QA Team) في المشروع**

بعد انتهاء المطورين من بناء التطبيق، يأتي دور فريق الاختبار وضمان الجودة (Quality Assurance - QA) للتأكد من أن التطبيق يعمل بدون مشاكل.

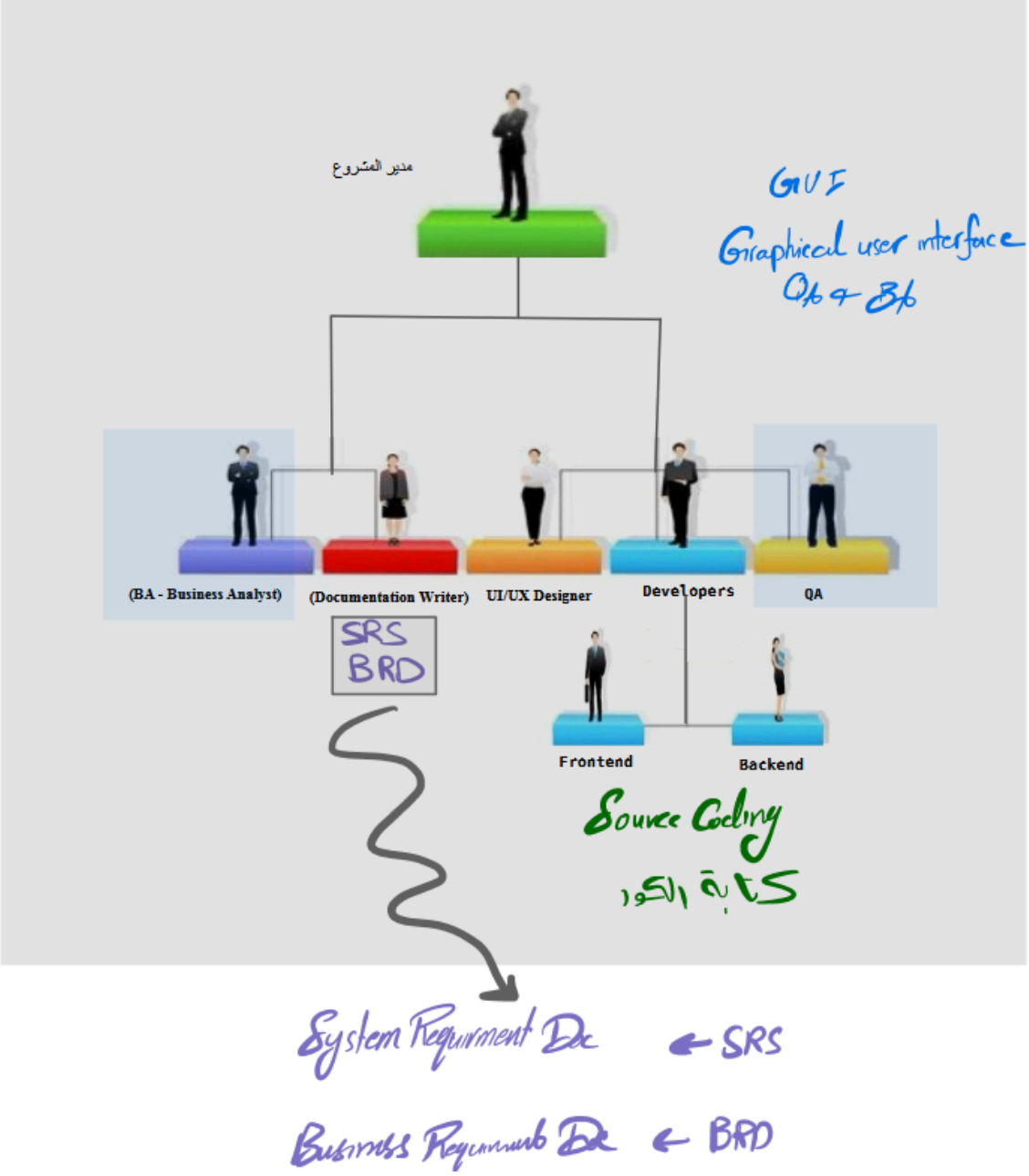
- التأكد من أن التطبيق يعمل كما هو مطلوب.
- اكتشاف الأخطاء (Bugs) قبل وصول المنتج للمستخدمين.
- اختبار الأداء والأمان وسرعة الاستجابة.
- التحقق من تجربة المستخدم (UX) وسهولة الاستخدام.

كيف يعمل فريق QA داخل المشروع؟

- يستلم النسخة التجريبية من المطورين.
- يبدأ الاختبار اليدوي والتلقائي.
- يكتب تقارير بالأخطاء ويرسلها للمطورين.
- يتم إصلاح الأخطاء وإعادة الاختبار.
- بعد التأكد من الجودة، يتم نشر التطبيق.



Hierarchy Positions



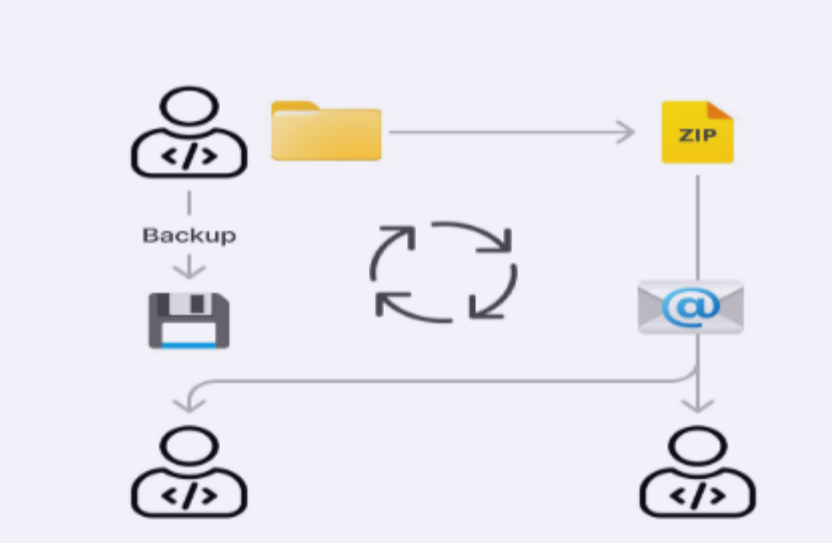
Task management tools

أدوات إدارة المهام (Task Management Tools) تساعد في تنظيم العمل وتحقيق أهداف المشروع. هناك العديد من الأدوات التي تستخدم لكل فريق:

1. **Project Manager (مدير المشروع):**
   - Jira / Trello
   - السبب: تساعد في إدارة المشروع، توزيع المهام، وتحديد الأولويات.
2. **GUI - Frontend & UI/UX Designers (مصممو واجهات المستخدم):**
   - Figma / Adobe XD + Trello
   - السبب: Figma أو Adobe XD لتصميم الواجهات، وTrello لإدارة المهام وتنظيم الأولويات.
3. **Source Code - Developers & Engineers (المطورين والمبرمجين):**
   - GitHub / GitLab / Jira
   - السبب: لإدارة الكود المصدر، تتبع الأخطاء (bugs)، ومراجعة المهام.
4. **Business Analysts (محللو الأعمال):**
   - Confluence / Notion
   - السبب: لتوثيق وتحليل المتطلبات، ومشاركة الأفكار مع الفريق.
5. **Documentation Writers (كتّاب الوثائق):**
   - أداة: Confluence / Google Docs
   - السبب: انشاء وتحرير الوثائق بسهوله، والتعاون مع الفريق.

كل فريق يتواصل مع باقي الفرق من خلال Slack أو Microsoft Teams لضمان سير العمل بكفاءة وتنظيم المهام.

الطرق التقليديه



- كل شخص يعمل على نسخة منفصلة من المشروع.
- مشاركة التعديلات تتم عبر ضغط الملفات ثم إرسالها عبر البريد الإلكتروني أو رفعها على Google Drive.
- يتم تكرار العملية بشكل مستمر حتى يتم إتمام المشروع.
- الحفاظ على نسخة احتياطية يتم على USB أو قرص خارجي في كل مرة يتم فيها حفظ نسخة من العمل يجب الخروج من البرنامج
- نتيجة لهذه العمليات، يتم الحصول على نسخ متعددة من المشروع.

المشاكل التي تنشأ:

- **تضييع الوقت**
- **صعوبة تتبع التعديلات**
- **التعاون صعب ومعقد**

SCM

الحل :

استخدام SCM:

SCM(Source Control Management) هو نظام يساعد في ادارة وتتبع التعديلات على الكود البرمجي .

يستخدمه المطورون لحفظ الاصدارات و العمل الجماعي و ومنع فقدان البيانات.

لماذا نستخدم SCM؟

- يحفظ كل التغييرات على الكود بشكل مرتب.
- يسمح لأكثر من شخص بالعمل على نفس المشروع بدون مشاكل.
- تقدر تعرف مين عدل بالكود ومتى وشو التعديلات.
- إذا صار خطأ، تقدر ترجع لأي إصدار قديم بسرعة.

Git

- هو **نظام إدارة وتحكم بالكود (SCM)**، يعني يساعد المطورين **يحفظوا، يتابعوا، ويرجعوا لأي إصدار من الكود** بسهولة.
- **موزّع (Distributed)**، يعني كل نسخة من المستودع عند كل مطور فيها **التاريخ الكامل للكود**، مش لازم يكون فيه خادم مركزي عشان يشتغل.

متى تستخدم Git؟

إذا كنت تشتغل على **مشروع فيه فريق** وتحتاج تتبع التعديلات بدون تعارض.
إذا كنت تحتاج **الرجوع لأي إصدار قديم** من الكود بسهولة.

GitHub

GitHub هو **منصة لاستضافة Git Repository** على الإنترنت، يعني مكان تقدر تخزن فيه الكود وتشاركه مع فريقك.

ليه بنستخدم GitHub؟

بيسهل التعاون بين **developers**، يعني لو في فريق شغال على نفس المشروع، كل واحد فيهم يقدر يشتغل على نسخته ويعمل تعديلات بتقدر ترفع الكود اللي تشتغل عليه، وتخلي باقي الفريق يراجعه ويعطي ملاحظات
وبيحفظ لك تاريخ كامل للتعديلات، يعني لو صار فيه مشكلة، تقدر ترجع لأي إصدار قديم من الكود.

متى نستخدم GitHub؟

لما نكون **مشتغلين مع فريق** على نفس المشروع وبدنا **نحفظ التعديلات** بشكل مرتب.

لما نحتاج **مراجعة الكود** من قبل **developers** تانيين قبل دمج التعديلات

لما نحتاج **تاريخ كامل للتعديلات** على الكود عشان نقدر نرجع لأي إصدار قديم إذا صار فيه مشكلة.

GitLab

GitLab هو **أداة لإدارة الأكواد** زي **GitHub**، لكن بيتميز إنه بيجمع مجموعة من الأدوات المتكاملة في مكان واحد. لو كنت شغال مع فريق على مشروع، GitLab بيخلي كل واحد يقدر يشتغل على نسخته الخاصة من الكود، وبعدها ندمج التعديلات مع بعض.

مميزاته:

إدارة الأكواد: بتقدر تخزن الأكواد وتتابع التعديلات اللي صارت عليها.
CI/CD: بيخليك تبني الكود، تختبره، وتنشره تلقائيًا على السيرفر.
مراجعة الكود: قبل ما تضيف أي تغيير، الفريق يقدر يراجعه أولاً.

إدارة المهام: تقدر تتابع التحديات أو المهام اللي لازم تتحل.
استضافة داخلية: إذا كنت حابب تحافظ على الأمان، تقدر تخلي GitLab على سيرفرك الخاص.

متى بنستخدم GitLab؟

- لو كنت شغال مع **فريق** على مشروع واحد وتحتاج **تنظيم** الأكواد، بالإضافة إنه بيساعدك في **اختبارات الكود** وبناء الكود بشكل تلقائي (CI/CD).
- إذا كنت **تحتاج استضافة داخلية** (على سيرفرات الشركة) لحماية خصوصية الكود
- كمان إذا كنت بحاجة لتحكم دقيق في مين ممكن يقدر يضيف أو يغير في الكود.

TFS

TFS أو **Team Foundation Server** هو **أداة** لإدارة الأكواد والمشاريع، شبيهة بـ **GitLab** و **GitHub**، لكن **TFS** تركيزها أكبر على **إدارة الفرق** والمشاريع داخل **الشركات**، وخصوصًا في بيئات **Microsoft**.

مميزاته:

إدارة الأكواد: بتخزن الأكواد وتتابع التعديلات اللي صارت عليها.
إدارة المهام والمشاريع: بيساعدك على متابعة التقدم في المشاريع وتنظيم المهام بين الفريق.
الاختبارات التلقائية: بيخلي بناء الكود واختباره أمر تلقائي.
الاستضافة داخلية: تقدر تستخدمه على سيرفرك الخاص بالشركة، أو باستخدام خدمة **Azure DevOps**
تتبع الأخطاء: بيساعد في تتبع الأخطاء وحلها بين أعضاء الفريق.

TFS هو نظام إدارة أكواد ومشاريع متكامل، وهو مثالي للشركات الكبيرة التي تحتاج إلى أدوات إدارة مشاريع وأكواد في بيئة واحدة، وخصوصًا إذا كانت تستخدم **تقنيات Microsoft**.

Azure

Azure هو **منصة سحابية** تقدمها شركة **مايكروسوفت**، وهي مش بس لإدارة الأكواد، لكن كمان بتوفر مجموعة كبيرة من الخدمات لتطوير، نشر، وإدارة التطبيقات والخدمات عبر الإنترنت.

مميزاته:

استضافة الأكواد: بتقدر تخزن أكوادك وتديرها، وبيدعم **Git** و **GitHub**.
CI/CD : توفر أدوات أتمتة لتطوير الكود، بناءه، واختباره بشكل تلقائي.
الخدمات السحابية: تقدر تستخدمها لتشغيل التطبيقات على السيرفرات بدون ما تحتاج للبنية التحتية المحلية.
إدارة التطبيقات: يمكن إدارة التطبيقات والخدمات عبر الإنترنت بشكل كامل، وتحسين أدائها.
أدوات تحليل البيانات: تقدم Azure خدمات تحليل البيانات والذكاء الاصطناعي.

باختصار، Azure هو **منصة سحابية** كاملة من مايكروسوفت بتساعدك في تطوير، بناء، ونشر التطبيقات على الإنترنت بشكل سهل ومرن، مع توفير كل الأدوات اللازمة لإدارة الأكواد والخدمات السحابية.