إليك ملخص شامل وكامل لكل ما قمنا ببنائه وتطويره في هذا المشروع **(منصة نظام إدارة البيئة)** من الصفر حتى التشغيل الكامل.

لقد قمنا بتحويل العملية اليدوية (Excel) إلى نظام ويب ديناميكي ومتكامل.

**1. نظرة عامة على النظام (Architecture)**

* **النوع:** تطبيق ويب (Web Application).
* **الخلفية (Backend):** Node.js + Express.js.
* **قاعدة البيانات (Database):** MySQL (يتم إدارتها عبر Sequelize ORM).
* **الواجهة (Frontend):** HTML + JavaScript (Vanilla) + Tailwind CSS للتصميم.
* **إدارة الملفات:** رفع الصور وتخزينها محلياً باستخدام مكتبة Multer.

**2. هيكلة قاعدة البيانات (Database Schema)**

قمنا بإنشاء 3 جداول رئيسية مترابطة:

1. **المستخدمون (Users):**
   * يخزن: الاسم، اسم المستخدم، كلمة المرور (مشفرة)، والدور (Role).
   * الأدوار: inspector (مفتش)، supervisor (مشرف)، env\_manager (مدير)، maintenance (صيانة).
2. **التقارير (Reports):**
   * يخزن: بيانات الزيارة الأساسية (الموقع، التاريخ، المفتش).
   * يخزن: **قائمة التحقق (Checklist)** كبيانات JSON (اجتياز/عدم اجتياز).
   * يخزن: **الحالة (Status)** التي تتحكم في سير العمل.
3. **الملاحظات (Observations):**
   * يخزن: تفاصيل المخالفات الفردية.
   * يخزن: صور المخالفة (قبل وبعد)، درجة الخطورة، المدة التصحيحية.
   * يخزن: الإجراء المتخذ وحالة الملاحظة (مفتوحة/مغلقة).

**3. سير العمل والإجراءات (Workflow Logic)**

هذا هو "العقل المدبر" للنظام الذي قمنا ببرمجته:

1. **المفتش (Inspector):**
   * يملأ النموذج الرقمي -> يرفع الصور -> يرسل الطلب.
   * حالة التقرير تصبح: pending\_supervisor (بانتظار المشرف).
2. **المشرف (Supervisor):**
   * يراجع التقرير -> يضغط "اعتماد".
   * حالة التقرير تصبح: pending\_manager (بانتظار المدير).
3. **مدير البيئة (Manager):**
   * يعتمد التقرير نهائياً.
   * حالة التقرير تصبح: approved\_for\_maintenance (معتمد - عند الصيانة).
4. **فريق الصيانة (Maintenance):**
   * يشاهد الملاحظات -> يقوم بالإصلاح -> يرفع صورة "بعد المعالجة".
   * حالة الملاحظة تصبح: تم التصحيح.

**4. الصفحات والواجهات التي تم بناؤها (Frontend)**

1. **login.html (صفحة الدخول):**
   * تصميم منقسم (Split Screen).
   * ربط مع API للتحقق من المستخدم وتخزين الـ Token.
2. **index.html (لوحة التحكم):**
   * عرض إحصائيات حية (عدد التقارير، الحالات).
   * جدول يعرض آخر التقارير مع ألوان مميزة لكل حالة.
   * تتكيف القوائم بناءً على اسم المستخدم وصلاحيته.
3. **inspection.html (نموذج التفتيش الرقمي):**
   * يحتوي على الأقسام الـ 10 للإكسل (النفايات، الضوضاء، إلخ).
   * إمكانية إضافة عدد لا نهائي من "بطاقات الملاحظات".
   * دعم رفع صور للمخالفات.
4. **view-report.html (صفحة التفاصيل والاعتماد):**
   * تعرض التقرير كاملاً (صور، ملاحظات، تشيك ليست).
   * تظهر أزرار "اعتماد/رفض" **فقط** للمشرف أو المدير.
   * تظهر نافذة منبثقة (Modal) **فقط** للصيانة لرفع صور الإصلاح.

**5. ملفات الكود الأساسية (Backend Files)**

* **server.js:** الملف الرئيسي لتشغيل الخادم، فتح المجلدات العامة، والاتصال بقاعدة البيانات.
* **src/config/db.config.js:** إعدادات الاتصال بـ MySQL.
* **src/models/index.js:** تجميع الجداول وتعريف العلاقات (Relations).
* **src/middleware/:**
  + authMiddleware.js: لحماية الصفحات (التحقق من الـ Token).
  + uploadMiddleware.js: لإدارة رفع الصور وتخزينها.
* **src/controllers/:**
  + authController.js: تسجيل الدخول وإنشاء المستخدمين.
  + reportController.js: حفظ التقرير، جلب البيانات، التعامل مع JSON والصور.
  + workflowController.js: إدارة تغيير الحالات (اعتماد، رفض، إغلاق ملاحظة).
* **seed.js:** ملف التشغيل الأولي لإنشاء الجداول وحسابات الموظفين.

**6. الحالة النهائية**

لديك الآن نظام **مؤتمت بالكامل**، يعمل على جهازك المحلي، يدعم تعدد المستخدمين والصلاحيات، ويحفظ البيانات والصور بشكل دائم.

**📋 تقرير إنجاز: منصة نظام إدارة البيئة (EMS)**

**1. بيانات الدخول للنظام (للتجربة)**

تم تهيئة قاعدة البيانات بـ 4 حسابات تمثل الدورة المستندية الكاملة للطلب.

كلمة المرور موحدة للجميع: 123456

| **اسم المستخدم** | **الدور (Role)** | **الصلاحيات والمهام** |
| --- | --- | --- |
| **inspector1** | مفتش بيئي ميداني | إنشاء التقارير، رفع الصور، والتحقق النهائي من الإصلاحات. |
| **section\_mgr** | مدير قسم البيئة | المراجعة الأولية لتقارير المفتشين (اعتماد / رفض). |
| **manager1** | مدير إدارة البيئة | الاعتماد النهائي للتقارير وتحويلها للصيانة. |
| **maint1** | فريق الصيانة | الاطلاع على الملاحظات، تنفيذ الإصلاح، ورفع صور الإثبات (إغلاق). |

**2. سير عمل الطلب (Workflow Cycle)**

تمت برمجة النظام ليعمل وفق التسلسل الهرمي التالي بدقة:

1. **إنشاء الطلب (المفتش):** يقوم المفتش بتعبئة النموذج ورفع الصور.
   * *الحالة:* طلب جديد -> بانتظار مدير القسم.
2. **المراجعة الأولية (مدير القسم):** يراجع الطلب.
   * *في حال الموافقة:* ينتقل إلى بانتظار مدير الإدارة.
   * *في حال الرفض:* يعود للمفتش وتصبح حالته مرفوض.
3. **الاعتماد النهائي (مدير الإدارة):** يراجع الطلب.
   * *في حال الموافقة:* ينتقل إلى لدى الصيانة.
   * *في حال الرفض:* يعود لمدير القسم.
4. **التنفيذ والمعالجة (الصيانة):** يقوم الفريق بإغلاق الملاحظات ورفع صور "بعد الإصلاح".
   * *الإجراء:* يتحول الطلب إلى تمت المعالجة (للمراجعة).
5. **الإغلاق والأرشفة (المفتش):** يدخل المفتش للتأكد من الإصلاح.
   * *الإجراء:* يعتمد الإصلاح ويصبح الطلب مكتمل (مؤرشف).

**3. ما تم إنجازه تقنياً وتصميمياً**

**أ) الواجهة الأمامية (Frontend)**

* **لوحة تحكم (Dashboard):** تعرض إحصائيات حية، وجدولاً بآخر الطلبات يوضح الحالة (نصياً باللون الأسود كما طلبت) مع زر تفاصيل.
* **نموذج تفتيش رقمي (Inspection Form):**
  + يحتوي على **10 أقسام** معيارية (النفايات، الضوضاء، المواد الكيميائية... إلخ).
  + خيارات التحقق: (اجتياز - أخضر) / (عدم اجتياز - أحمر) / (لا ينطبق - رمادي).
  + إمكانية إضافة **ملاحظات لا نهائية** مع الصور، تحديد الخطورة، والمدة الزمنية.
  + هيدر ثابت (Sticky Header) لسهولة التنقل.
* **صفحة التفاصيل (Report Details):**
  + تعرض البيانات بشكل "للقراءة فقط".
  + **تحسين بصري للقائمة:** الخيار الذي حدده المفتش يظهر بارزاً وملوناً، وباقي الخيارات باهتة.
  + **عارض صور (Lightbox):** عند النقر على الصورة تظهر مكبرة.
  + **أزرار ديناميكية:** تظهر أزرار (اعتماد/رفض) أو (إصلاح) حسب دور المستخدم وحالة الطلب فقط.

**ب) الخلفية البرمجية (Backend - Node.js)**

* **قاعدة البيانات (MySQL & Sequelize):** بناء جداول مترابطة (Users, Reports, Observations) تدعم التوسع.
* **نظام الحماية (Auth):** استخدام JWT Tokens لتأمين الجلسات وتشفير كلمات المرور بـ Bcrypt.
* **إدارة الملفات (Multer):** نظام لرفع وتخزين الصور محلياً في مجلد uploads.
* **وحدة تحكم سير العمل (Workflow Controller):** منطق ذكي يدير تغيير الحالات ويمنع المستخدمين من اتخاذ إجراءات خارج صلاحياتهم.

**4. كيف تبدأ العمل الآن؟**

النظام جاهز تماماً على جهازك. خطوات التشغيل الروتينية هي:

1. تأكد من تشغيل **XAMPP** (خدمة MySQL).
2. افتح المجلد في VS Code وافتح التيرمينال.
3. شغل الخادم بالأمر:

Bash

node server.js

1. افتح المتصفح على الرابط: http://localhost:3000