تقرير فني شامل - نظام حماية البيانات المتكامل

Comprehensive Technical Report - Integrated Data Protection System

☐ جدول المحتويات / Table of Contents

1. المشاكل التي واجهناها وكيف تم حلها

1.1 مشكلة الاعتماد على نماذج Spacy في Presidio

1.1.1 وصف المشكلة

خلال مرحلة تطوير النظام، واجهنا تحديا تقنيا يتعلق بالاعتماد على مكتبة Microsoft Presidio مكتبة (Pre-trained Models) مثل Spacy لاكتشاف البيانات الحساسة في النصوص. يعتمد Presidio على لتوفير قدرات معالجة اللغة الطبيعية (NLP) المتقدمة. وخاصة النماذج المدربة مسبقا

بين البيانات الحساسة الحقيقية والبيانات العادية NLP يستخدم تقنيات Presidio - السياق التقني:
على سبيل المثال، يميز بين رقم عادي مثل "12345" ورقم هاتف حقيقي مثل "050-123-4567 للتمييز
من فهم السياق والأنماط اللغوية Presidio توفر النماذج اللغوية المدربة التي تمكن Spacy - "-

واجهنا ثلاث تحديات رئيسية: طبيعة المشكلة:

تعقيد عملية التثبيت: .1

نماذج Spacy كبيرة الحجم (تتراوح بين 100-500 ميجابايت) .1

تتطلب عملية تحميل منفصلة عبر الأمر: .1

في بيئات Windows، قد تفشل عملية التحميل بسبب مشاكل في الصلاحيات أو الاتصال بالإنترنت .1

1.

في بيئات محدودة الموارد (مثل الخوادم السحابية الصغيرة)، قد لا تتوفر مساحة تخزين كافية

1.

الاعتماد على بيئة محددة:

النماذج مرتبطة بإصدارات محددة من Python و Spacy .1

قد لا تعمل بشكل صحيح في بيئات مختلفة (Docker containers، Cloud environments، إلخ) .1

1.

بعض النماذج قد لا تكون متاحة لجميع اللغات المطلوبة

1.

تأثير على استقرار النظام:

عند عدم توفر نماذج Spacy، Presidio يرمي استثناء (Exception) ويتوقف .1

النظام كان يفشل بالكامل عند محاولة استخدام Presidio .1

هذا يمنع المستخدمين من استخدام النظام حتى في حالة تثبيت Presidio نفسه .1

1.1.2 الحل المطبق

حالة عدم توفر المكونات الاختيارية.Graceful Degradationلحل هذه المشكلة، قمنا بتطبيق استراتيجية (التدهور التدريجي) التي تسمح للنظام بالعمل حتى في

تم تصميم نظام متعدد المستويات (Multi-tier Fallback System) يتكون من ثلاث طبقات: النهج المتبع:

الطبقة الأولى: محاولة استخدام Presidio مع Spacy .1

الطبقة الثانية: استخدام Presidio بدون Spacy (إن أمكن) .1

الطبقة الثالثة: استخدام أنماط Regex كبديل نهائي .1

التنفيذ التقني:

18-12)، قمنا بإضافة معالجة استثناءات عند استيراد Presidio:في ملف أ) معالجة عدم توفر Presidio: (السطور

منطق للتحقق من وجود النماذج ومحاولة استخدام نماذج بديلة: ب) معالجة عدم توفر نماذج Spacy: في نفس الملف (السطور 32-44)، أضفنا

نظام Regex:عند فشل تهيئة Presidio (السطور 65-69)، يتم تعيين ج) نظام Fallback عند الفشل: والاعتماد على

(السطور 73-110) تستخدم أنماط Regex محددة لاكتشاف:تم تطوير دالة د) نظام Regex البديل: - أرقام بطاقات الائتمان: - عناوين البريد الإلكتروني: - أرقام الهواتف:

158-127)، يتم التحقق من توفر Presidio قبل الاستخدام:في دالة هـ) منطق الاختيار التلقائي: (السطور

1.1.3 النتائج والتقييم

النظام يعمل بشكل كامل حتى في حالة عدم توفر Presidio أو Spacyالمرونة: ☐ .1 النتائج المحققة: النشر:3. ☐ لا يتوقف النظام عند فشل التثبيت، بل ينتقل تلقائيا للنظام البديلالاستقرار:2. ☐ المستخدم:4. ☐ يمكن نشر النظام في بيئات مختلفة دون قلق من متطلبات التثبيت المعقدةسهولة المستخدمون يمكنهم استخدام النظام مباشرة دون الحاجة لإعدادات إضافيةتجربة

في اكتشاف البيانات الحساسة (حوالي 70-80% مقابل 90-95%)الدقة: - المقايضة (Trade-off): أسرع في المعالجة لكن أقل ذكاء في فهم السياق Regex الأداء:- نظام Regex أقل دقة من Presidio

النظام في جميع الحالات مع الحفاظ على القدرة على الاستفادة من Presidio عند توافره. الخلاصة: هذا الحل يوفر توازنا جيدا بين المرونة والوظيفية، حيث يضمن عمل

1.2 مشكلة الاعتماد على خدمة MyDLP الخارجية

1.2.1 وصف المشكلة

عبر قنوات مختلفة (البريد الإلكتروني، الويب، USB، إلخ).MyDLPفي تصميم النظام، قمنا بدمج قرارات المنع الفعلية. يعمل MyDLP كطبقة تنفيذية (Enforcement Layer) تمنع نقل البيانات الحساسة (نظام منع فقدان البيانات مفتوح المصدر) كأداة لتنفيذ

المحددة:Policy Service- يكتشف البيانات الحساسة في النصوص:Presidio - السياق المعماري: ينفذ قرارات المنع فعليا على مستوى الشبكة:MyDLP- يقرر الإجراء المطلوب بناء على السياسات

واجهنا ثلاث تحديات رئيسية: طبيعة المشكلة:

تعقيد التثبيت والإعداد: 1.

(إلخ ،MySQL/MariaDB ،Erlang/OTP) يتطلب تثبيت عدة مكونات MyDLP 1.

يحتاج إعداد خادم منفصل يعمل على منفذ 8080 1.

في بيئة التطوير المحلية، قد لا يكون التثبيت مرغوبا أو متاحا 1.

1.

يتطلب معرفة تقنية متقدمة لإعداده بشكل صحيح

1.

الاعتماد على خدمة خارجية:

النظام يتواصل مع MyDLP عبر HTTP API 1.

عند فشل الاتصال، النظام كان يرمي استثناء ويتوقف تماما 1.

هذا يمنع اختبار وتطوير النظام بدون تثبيت MyDLP الكامل 1.

1.

في بيئات الإنتاج، قد يكون MyDLP على خادم منفصل مما يزيد من احتمالية فشل الاتصال

1.

تأثير على دورة التطوير:

المطورون لا يستطيعون تطوير أو اختبار النظام بدون تثبيت MyDLP 1.

هذا يبطئ عملية التطوير ويجعلها معقدة 1.

في بيئات CI/CD، قد لا يكون تثبيت MyDLP عمليا 1.

1.2.2 الحل المطبق

لحل هذه المشكلة، قمنا بتطبيق نمط Simulation Mode.MyDLP بشكل كامل حتى في حالة عدم توفر (وضع المحاكاة) الذي يسمح للنظام بالعمل

دعم كامل لوضع المحاكاة (Fully Optional) اختيارية تماماتم تصميم خدمة MyDLP لتكون النهج المتبع: مع:

فحص ما إذا كانت MyDLP مفعلة ومتاحةالتحقق من حالة الخدمة: 1.

عدم رفع استثناءات عند فشل الاتصالمعالجة الأخطاء بشكل هادئ: 1.

إرجاع نتائج محاكية عند عدم توفر الخدمةمحاكاة السلوك المطلوب: .1

التنفيذ التقني:

(السطور 15-29):في ملف أ) تهيئة الخدمة مع دعم الوضع المحاكي:

79-43)، قمنا بمعالجة جميع أنواع الأخطاء بشكل هادئ:في دالة ب) معالجة طلبات API مع Fallback: (السطور

(السطور 81-116):في دالة ج) دالة المنع مع وضع المحاكاة:

(السطور 143-173).نفس النهج مطبق في د) دالة منع البريد مع وضع المحاكاة:

1.2.3 النتائج والتقييم

☐ المطورون يمكنهم تطوير واختبار النظام بدون تثبيت MyDLPاستقلالية التطوير: ☐ .1 النتائج المحققة: الخدمة:3. ☐ يمكن نشر النظام في بيئات مختلفة مع أو بدون MyDLPالمرونة في النشر:2. في الكودسهولة التكامل:4. ☐ النظام يستمر في العمل حتى عند فشل الاتصال بـ MyDLPاستمرارية يمكن تفعيل MyDLP لاحقا بدون أي تغييرات

نقل البيانات فعليا، لكن يتم تسجيل الأحداث وإنشاء التنبيهاتالأمان الفعلي: - المقايضة (Trade-off): يجب توثيق أن النظام في وضع محاكاة للمستخدمينالوضوح:- في وضع المحاكاة، لا يتم منع

كبيرة في التطوير والنشر مع الحفاظ على إمكانية الاستفادة الكاملة من MyDLP عند توافره. الخلاصة: هذا الحل يوفر مرونة

1.3 مشكلة استخراج النص من الملفات متعددة الصيغ

1.3.1 وصف المشكلة

معالجة الملفات مباشرة، كان من الضروري تطوير آلية لاستخراج النص من هذه الملفات قبل التحليل. PDF، DOCX، لاكتشاف البيانات الحساسة. نظرا لأن Presidio يعمل على النصوص فقط ولا يمكنه أحد المتطلبات الأساسية للنظام هو القدرة على تحليل محتوى الملفات المختلفة (XLSX

ملف → استخراج النص → Presidio → اكتشاف البيانات الحساسةسير العمل: - السياق المعماري:
دعم صيغ متعددة (PDF، DOCX، XLSX، TXT)المتطلبات:-

واجهنا ثلاث تحديات رئيسية: طبيعة المشكلة:

تعدد المكتبات المطلوبة: .1

كل نوع ملف يحتاج مكتبة مختلفة: .1

أو :PDF .2

DOCX: .2

XLSX: .2

XLS: .2

.1

هذا يزيد من تعقيد التبعيات (Dependencies)

.1

مشاكل التثبيت:

المستخدم قد لا يثبت جميع المكتبات المطلوبة .1

يحتاج مكتبات إضافية)بعض المكتبات كبيرة الحجم (مثل .1

.1

في بيئات محدودة الموارد، قد لا يكون تثبيت جميع المكتبات مرغوبا

.1

تأثير على موثوقية النظام:

عند رفع ملف بدون المكتبة المطلوبة، النظام كان يرمي استثناء ويتوقف .1

هذا يمنع المستخدمين من رفع الملفات حتى لو كانت صيغ أخرى مدعومة .1

لا يوجد نظام بديل عند فشل استخراج النص .1

1.3.2 الحل المطبق

التي تدعم أكثر من مكتبة.Multiple Library Supportالحل هذه المشكلة، قمنا بتطبيق استراتيجية (دعم متعدد للمكتبات) مع نظام Fallback للملفات

شاملة للأخطاء:2. خاصة لملفات PDFدعم مكتبات متعددة لنفس الصيغة: .1 النهج المتبع:
سهولة إضافة صيغ جديدةتصميم قابل للتوسع:3. رسائل واضحة عند عدم توفر المكتباتمعالجة

التنفيذ التقني:

14-23)، قمنا بإنشاء قاموس يربط كل صيغة بدالة الاستخراج المناسبة:في ملف أ) هيكل الخدمة: (السطور

72-121)، قمنا بتطبيق نظام Fallback متعدد المستويات:في دالة ب) استخراج PDF مع دعم مكتبتين: (السطور

DOCX و XLSX (السطور 123-190)، قمنا بتوفير رسائل خطأ واضحة: ج) معالجة الأخطاء للصيغ الأخرى: لصيغ

(السطور 25-51)، قمنا بإضافة التحقق من الصيغة المدعومة:في دالة د) دالة الاستخراج الرئيسية:

1.3.3 النتائج والتقييم

☐ دعم مكتبات متعددة لنفس الصيغة يزيد من احتمالية نجاح الاستخراجالمرونة: ☐ .1 النتائج المحققة: للتوسع:3. ☐ رسائل خطأ واضحة تساعد المستخدمين على تثبيت المكتبات المطلوبةالوضوح:2.
بالكامل عند فشل استخراج ملف واحدالموثوقية:4. ☐ سهولة إضافة صيغ جديدة أو مكتبات بديلةالقابلية
النظام لا يتوقف

العملية قليلاالأداء:- دعم مكتبات متعددة يزيد من تعقيد الكود قليلاالتعقيد: - المقايضة (Trade-off):
محاولة استخدام مكتبات متعددة قد يبطئ

يوفر موثوقية عالية في استخراج النص من الملفات مع الحفاظ على المرونة والقابلية للتوسع. الخلاصة:
هذا الحل

1.4 اختيار نظام إدارة قواعد البيانات

1.4.1 وصف المشكلة

المكتشفة. كان من الضروري اختيار نظام إدارة قواعد بيانات مناسب يوفر الأداء والموثوقية المطلوبة.
إلى تخزين بيانات متنوعة تشمل معلومات المستخدمين، السياسات، التنبيهات، السجلات، والكيانات
النظام يحتاج

المطلوبةالسياسات:- معلومات المستخدمين والصلاحيات والجلساتالمستخدمون: - البيانات المخزنة:
تنبيهات اكتشاف البيانات الحساسة مع التفاصيلالتنبيهات:- قواعد حماية البيانات والإجراءات
الحساسة المكتشفة (مشفرة)الكيانات المكتشفة:- سجل شامل لجميع الأحداث والأنشطةالسجلات:-
البيانات

قاعدة البيانات عبر متغير البيئة - استخدام SQLAlchemy كـ ORM للتفاعل مع قاعدة البيانات المتطلبات:
- تحديد نوع

واجهنا تحديا في اختيار نظام قاعدة البيانات المناسب: طبيعة المشكلة:

متطلبات PostgreSQL: 1.

يتطلب تثبيت خادم PostgreSQL منفصل 1.

يحتاج إعداد قاعدة بيانات ومستخدم مع الصلاحيات المناسبة 1.

يجب أن يكون الخادم يعمل قبل تشغيل النظام 1.

1.

في بيئة التطوير المحلية، قد لا يكون PostgreSQL مثبتا أو متاحا

1.

متطلبات الإنتاج:

1. يحتاج نظام قاعدة بيانات قوي وموثوق

1. يجب دعم الاتصالات المتزامنة (Concurrent Connections)

1. يحتاج Connection Pooling للأداء الأمثل

1.4.2 الحل المطبق

بعد دراسة الخيارات المتاحة، قمنا باختيار PostgreSQL. محسن للنظام، مع تطبيق Connection Pooling كنظام قاعدة بيانات أساسي

مبررات الاختيار: 1. الأداء: PostgreSQL يوفر أداء ممتازا مع البيانات الكبيرة 2. الموثوقية: سلامة البيانات 3. الميزات المتقدمة: دعم متقدم لـ JSON، Full-text search، والميزات المتقدمة 4. المرونة: نظام ACID-compliant يضمن سواء للإنتاج والتطوير على حد مناسب

التنفيذ التقني:

(السطور 24-28):في ملف أ) إعداد الاتصال:

(السطور 10-18):في ملف ب) إعداد محرك قاعدة البيانات:

تم تطبيق Connection Pooling لتحسين الأداء: ج) Connection Pooling: نشطة دائما:pool_size=10- يسمح بزيادة الاتصالات حتى 30 عند الحاجة:max_overflow=20- يحافظ على 10 اتصالات
يتحقق من صحة الاتصال قبل الاستخدام:pool_pre_ping=True-

1.4.3 النتائج والتقييم

النتائج المحققة: 1. الأداء: Connection Pooling يحسن الأداء بشكل كبير 2. ☐ الموثوقية: PostgreSQL يوفر موثوقية عالية لبيانات الإنتاج 3. ☐ المرونة: استخدام نفس الإعدادات للتطوير والإنتاج يدعم PostgreSQL التوسع الأفقي والعمودي القابلية للتوسع: 4. ☐ يمكن

يحتاج إعداد قاعدة بيانات ومستخدم - □□ يتطلب تثبيت وتشغيل PostgreSQL قبل البدء المتطلبات:
- □□

اختيار PostgreSQL يوفر أساسا قويا وموثوقا للنظام مع ضمان الأداء والموثوقية المطلوبة. الخلاصة:

1.5 مشكلة اكتشاف مسارات الملفات الثابتة

1.5.1 وصف المشكلة

FastAPI .النظام يحتوي على واجهة مستخدم ويب (HTML، CSS، JavaScript) موجودة في مجلد
.يحتاج إلى ربط هذه الملفات بشكل صحيح لتقديمها للمستخدمين عند الوصول إلى النظام

يحتاج معرفة المسار الصحيح للملفات ، يجب عرض الواجهة- عند فتح - الملفات موجودة في: السياق:
FastAPI -

واجهنا مشكلة في اكتشاف المسار الصحيح للملفات الثابتة: طبيعة المشكلة:

اعتماد المسارات على موقع التشغيل: 1.

، المسار النسبي مختلفعند التشغيل من 1.

عند التشغيل من جذر المشروع، المسار مختلف 1.

في بيئة Docker، المسارات مختلفة تماما 1.

1.

في بيئات Cloud، قد تكون المسارات مختلفة

1.

فشل في العثور على الملفات:

يفشل عند تغيير موقع التشغيلاستخدام مسارات ثابتة مثل 1.

المستخدم يرى خطأ 404 بدلا من الواجهة .1

1.

هذا يؤثر سلبا على تجربة المستخدم

1.

عدم المرونة في النشر:

لا يمكن نشر النظام في بيئات مختلفة بسهولة .1

يتطلب تعديل الكود لكل بيئة .1

1.5.2 الحل المطبق

بناء على موقع ملف الكود الحالي.Dynamic Path Discoveryلحل هذه المشكلة، قمنا بتطبيق نظام (اكتشاف المسار الديناميكي) الذي يحسب المسار الصحيح

متعددة:2. لحساب المسار النسبي استخدام حساب المسار بناء على موقع الملف: .1 النهج المتبع: الملفاتمعالجة الأخطاء بشكل هادئ:3. محاولة مسارات بديلة عند فشل المسار الأولدعم مسارات النظام يستمر في العمل حتى عند فشل العثور على

التنفيذ التقني:

(السطور 61-89):في ملف أ) اكتشاف المسار الديناميكي:

في نفس الملف (السطور 136-164): ب) دالة الجذر مع اكتشاف ديناميكي:

1.5.3 النتائج والتقييم

النظام يعمل من أي موقع (جذر المشروع، (Cloud ،Docker ،backend/المرونة: ☐ .1 النتائج المحققة: الثابتةالاستمرارية:3. ☐ دعم مسارات متعددة يزيد من احتمالية العثور على الملفاتالموثوقية:2. ☐ بدون تعديلاتسهولة النشر:4. ☐ النظام يستمر في العمل حتى عند فشل العثور على الملفات يمكن نشر النظام في بيئات مختلفة

في كل طلب قد يبطئ العملية قليلا (يمكن تحسينه بالـ Caching)الأداء: - المقايضة (Trade-off):
الكود أكثر تعقيدا قليلا لكنه يوفر مرونة أكبرالتعقيد:- حساب المسار

مرونة عالية في اكتشاف الملفات الثابتة مع ضمان عمل النظام في جميع البيئات المختلفة. الخلاصة:
هذا الحل يوفر

1.6 ملخص المشاكل والحلول

الخلاصة العامة

مرحلة تطوير النظام، واجهنا خمس مشاكل رئيسية تم حلها بنجاح باستخدام استراتيجيات مختلفة:
خلال

1. باستخدام نظام Fallback متعدد المستويات (Graceful Degradation)مشكلة الاعتماد على Spacy: تم حلها

1. تم حلها باستخدام وضع المحاكاة (Simulation Mode)مشكلة الاعتماد على MyDLP:

1. تم حلها باستخدام دعم متعدد للمكتبات (Multiple Library Support)مشكلة استخراج النص:

1. تم اختيار PostgreSQL مع Connection Pooling محسناختيار قاعدة البيانات:

1. حلها باستخدام اكتشاف المسار الديناميكي (Dynamic Path Discovery)مشكلة مسارات الملفات: تم

الدروس المستفادة

1. تصميم النظام ليعمل حتى في حالة عدم توفر المكونات الاختياريةالمرونة في التصميم:

1. عدم رفع استثناءات عند فشل المكونات الاختياريةمعالجة الأخطاء بشكل هادئ:

1. توفير أنظمة بديلة عند فشل النظام الأساسيدعم البدائل:

استخدام المسارات الديناميكية بدلا من الثابتةاكتشاف ديناميكي: .1

التأثير على جودة النظام

تجربة المطورين:- ☐ النظام يعمل في جميع الحالاتزيادة الموثوقية:- ☐ الحلول المطبقة ساهمت في:
☐ يمكن نشر النظام في بيئات مختلفةسهولة النشر:- ☐ سهولة التطوير والاختبارتحسين
النظام لا يتوقف عند فشل المكونات الاختياريةالاستمرارية:-

2. كيف تم بناء المشروع

2.1 البنية المعمارية (Architecture)

مع فصل واضح للطبقات:معمارية FastAPIالمشروع مبني على

2.2 خطوات البناء

الخطوة 1: إعداد البيئة الافتراضية

الخطوة 2: تثبيت التبعيات

: تحليل النصوص - لقاعدة البيانات ORM :- : خادم ASGI- : إطار عمل API - التبعيات الرئيسية:
: استخراج النص من الملفات , ,- : التشفير-

الخطوة 3: إعداد قاعدة البيانات

الخطوة 4: تشغيل التطبيق

2.3 نماذج قاعدة البيانات

تم إنشاء النماذج التالية:

1. User (المستخدمون):

1. Policy (السياسات):

1. Alert (التنبيهات):

1. Log (السجلات):

1. DetectedEntity (الكيانات المكتشفة):

2.4 نظام المصادقة

تم تنفيذ نظام مصادقة باستخدام JWT:

3. إضافة أداتي MyDLP و Presidio

3.1 موقع كود Presidio

الملف الرئيسي:

الوظائف الرئيسية:

1. Presidio Analyzer تهيئة :

1. تحليل النص واكتشاف البيانات الحساسة :

1. Regex نظام Fallback باستخدام :

1. فحص وجود بيانات حساسة :

1. الحصول على أنواع الكيانات المدعومة :

:الإعدادات في

3.2 موقع كود MyDLP

الملف الرئيسي:

الوظائف الرئيسية:

: تهيئة خدمة MyDLP 1.

: منع نقل البيانات 1.

: منع إرسال البريد 1.

: مراقبة حركة الشبكة 1.

: إنشاء سياسة في MyDLP 1.

: إرسال طلبات API إلى MyDLP 1.

:الإعدادات في

3.3 التكامل بين الأداتين

الملف الرئيسي:

هذا الملف يدمج Presidio و MyDLP:

4. شرح كيفية عمل الأداتين مع الكود

4.1 سير العمل الكامل (Complete Workflow)

4.2 مثال عملي: تحليل نص

:الكود في

ما يحدث بالتفصيل:

:يحلل النص Presidio 1.

:يطبق القواعد Policy Service 1.

:يمنع النقل MyDLP 1.

4.3 مثال عملي: تحليل ملف

:الكود في

4.4 مثال عملي: مراقبة البريد الإلكتروني

:الكود في

5. شرح طريقة تحليل الملفات

5.1 أنواع الملفات المدعومة

النظام يدعم الأنواع التالية:

- ملفات النص العاديTXT 1.

- مستندات PDFPDF 1.

- مستندات Word (2007+)DOCX 1.

- جداول Excel (2007+)XLSX .1

- مستندات Word القديمة (غير مدعومة مباشرة)DOC .1

- جداول Excel القديمة (تتطلب xlrd)XLS .1

5.2 آلية الاستخراج

الملف:

5.2.1 استخراج من TXT

- دعم الملفات في الذاكرة أو من القرص - معالجة أخطاء الترميز - دعم ترميزات متعددة الميزات:

5.2.2 استخراج من PDF

الملفات في الذاكرة - استخراج من جميع الصفحات - دعم مكتبتين (pdfplumber و PyPDF2) الميزات:
- دعم

5.2.3 استخراج من DOCX

- دعم الملفات في الذاكرة - الحفاظ على البنية - استخراج من الفقرات والجداول الميزات:

5.2.4 استخراج من XLSX

للحصول على القيم فقط- استخدام - الحفاظ على بنية الجدول - استخراج من جميع الأوراق الميزات:

5.3 سير العمل الكامل لتحليل الملف

5.4 مثال عملي كامل

مستخدم يرفع ملف PDF يحتوي على رقم هاتفسيناريو:

5.5 معالجة الأخطاء

النظام يتعامل مع الأخطاء التالية:

1. نوع ملف غير مدعوم:

1. فشل استخراج النص:

1. ملف فارغ:

6. الخلاصة

6.1 ما تم إنجازه

☐ نظام متكامل يجمع بين Presidio و MyDLP

☐ دعم متعدد للملفات (PDF, DOCX, XLSX, TXT)

☐ نظام Fallback يعمل حتى بدون Presidio أو MyDLP

☐ واجهة API كاملة مع توثيق تلقائي

☐ نظام مصادقة باستخدام JWT

☐ قاعدة بيانات مع Migrations

☐ نظام تنبيهات شامل

☐ تشفير البيانات الحساسة

6.2 الملفات الرئيسية

النص من الملفاتخدمة MyDLP للمنعخدمة Presidio للتحليلالوظيفةالملف
نقطة الدخول الرئيسيةواجهات API للتحليلتطبيق السياساتاستخراج

6.3 الخطوات التالية

1. إضافة Caching للتحليلات المتكررةتحسين الأداء:

1. تحسين دعم اللغة العربية في Presidioدعم لغات إضافية:

1. إضافة المزيد من الميزاتواجهة مستخدم محسنة:

1. إضافة تقارير تفصيليةتقارير متقدمة:

1. ربط النظام مع أنظمة SIEMتكامل مع SIEM:

2024 آخر تحديث:

1.0.0 الإصدار: