

تقرير مشروع: استوديو الصور (Photo Studio)

مقدم إلى: وزارة الدفاع فرع الذكاء الاصطناعي

مقدم من:

- الاسم: حامد محمد المرعي
- الرقم الجامعي/التعريف: 2223136

تاريخ التقديم: 20 يوليو 2025

جدول المحتويات

- المقدمة
- متطلبات النظام 2.1، المتطلبات الوظيفية 2.2، المتطلبات غير الوظيفية
- هندسة النظام 3.1، المكونات الرئيسية 3.2، تدفق البيانات
- الميزات والوظائف 4.1، إدارة المستخدمين 4.2، رفع ومعاينة الصور 4.3، الفلاتر الكلاسيكية 4.4، فلاتر الذكاء الاصطناعي (AI)
- التنفيذ التقني 5.1، الواجهة الخلفية (Backend) 5.2، الواجهة الأمامية (Frontend) 5.3، قاعدة البيانات والتخزين 5.4، النشر 5.5 (Deployment) دعم تعدد اللغات والوضع الليلي/النهارى
- التحديات والحلول
- العمل المستقبلي والتحسينات
- الخاتمة
- المراجع
- الملاحق

1. المقدمة

يهدف هذا التقرير إلى توثيق مشروع "استوديو الصور"، وهو تطبيق ويب يتيح للمستخدمين معالجة الصور رقميًا من خلال مجموعة متنوعة من الفلاتر الكلاسيكية وفلاتر الذكاء الاصطناعي. في عصر يتزايد فيه الاعتماد على المحتوى البصري، أصبح تعديل الصور ضرورة للمصممين والمستخدمين العاديين على حد سواء. يوفر هذا المشروع منصة سهلة الاستخدام وقوية لتحويل الصور الخام إلى أعمال فنية جذابة، مع التركيز على الأداء، تجربة المستخدم، والمرونة.

يتناول التقرير تفاصيل متطلبات النظام، الهندسة المعمارية، الميزات المطبقة، التقنيات المستخدمة في التنفيذ، التحديات التي واجهت عملية التطوير وكيفية التغلب عليها، بالإضافة إلى الخطط المستقبلية للتحسين.

2. متطلبات النظام

2.1. المتطلبات الوظيفية

- المصادقة وإدارة المستخدمين:
 - تسجيل مستخدمين جدد.
 - تسجيل دخول وخروج آمن.
 - إدارة ملفات تعريف المستخدمين (مثل تغيير كلمة المرور).
- رفع الصور:
 - إمكانية رفع الصور بتنسيقات شائعة (JPG, PNG).
 - التحقق من صحة ملفات الصور.
- تطبيق الفلاتر:
 - عرض قائمة بالفلاتر الكلاسيكية وفلاتر الذكاء الاصطناعي بشكل منفصل.

- تطبيق الفلاتر المختارة على الصورة المرفوعة.
- معاينة فورية لتأثير الفلتر.
- **الفلاتر الكلاسيكية:**
 - قص الصورة (Crop) مع أداة قص تفاعلية.
 - تحويل الصورة إلى الرمادي (Grayscale).
 - قلب ألوان الصورة (Invert Colors).
 - زيادة حدة الصورة (Sharpen) مع تحكم في مستوى الحدة.
- **فلاتر الذكاء الاصطناعي (AI):**
 - طمس الوجوه (Blur Faces) مع تحكم في مستوى التعتيم.
 - التعرف على وجوه البشر (Human Face Detection) وتحديدتها.
 - التعرف على وجوه القطط (Cat Face Detection) وتحديدتها.
 - التعرف على العيون (Eye Detection) وتحديدتها.
 - التعرف على الابتسامات (Smile Detection) وتحديدتها.
 - التعرف على كامل الجسد (Full Body Detection) وتحديدتها.
 - التعرف على الجزء العلوي من الجسد (Upper Body Detection) وتحديدتها.
 - التعرف على لوحات السيارات (License Plate Detection) وتحديدتها.
 - مستقبلاً حفظ مجموعة من الوجوه وإمكانية التعرف عليها.
- **حفظ الصور المعالجة:**
 - حفظ الصورة بعد تطبيق الفلتر في حساب المستخدم.
 - تخزين الصور في خدمة تخزين سحابية.
- **إدارة الصور المحفوظة:**
 - عرض جميع الصور المعالجة الخاصة بالمستخدم.
 - عرض تفاصيل كل صورة (الفلتر المطبق، تاريخ المعالجة).
 - حذف الصور المحفوظة.

2.2. المتطلبات غير الوظيفية

- **الأداء:**
 - سرعة معالجة الصور ومعاينتها.
 - استجابة سريعة لواجهة المستخدم.
- **قابلية التوسع:**
 - القدرة على التعامل مع زيادة عدد المستخدمين والصور.
 - استخدام خدمات سحابية قابلة للتوسع.
- **الأمان:**
 - تأمين بيانات المستخدمين والصور.
 - حماية ضد الثغرات الأمنية الشائعة مثل (CSRF).
- **سهولة الاستخدام (Usability):**
 - واجهة مستخدم بديهية وسهلة التنقل.
 - دعم تعدد اللغات (العربية والإنجليزية).
 - دعم الوضع الليلي والنهاري.
- **الاعتمادية:**
 - ضمان استمرارية الخدمة وتقليل وقت التوقف.
- **الاستجابة (Responsiveness):**
 - تصميم متكيف يعمل بشكل جيد على مختلف أحجام الشاشات (الهواتف الذكية، الأجهزة اللوحية، أجهزة الكمبيوتر المكتبية).

3. هندسة النظام

يعتمد مشروع "استوديو الصور" على بنية تطبيق ويب ثلاثية الطبقات (Three-Tier Architecture) ، تتكون من طبقة العرض (Frontend) ، طبقة المنطق (Backend) ، وطبقة البيانات (Database & Storage).

3.1. المكونات الرئيسية

- **الواجهة الأمامية: (Frontend)**
 - **HTML5:** لهيكل الصفحة.
 - **CSS3 (Tailwind CSS):** للتصميم والاستجابة.
 - **JavaScript:** للتفاعل مع المستخدم، المعاينات الحية، وأدوات التعديل (مثل Cropper.js).
- **الواجهة الخلفية: (Backend)**
 - **Django (Python):** إطار عمل الويب الرئيسي، يدير منطق الأعمال، مسارات URL ، ونماذج البيانات.
 - **OpenCV (cv2):** مكتبة معالجة الصور الرئيسية، تستخدم لتطبيق الفلاتر الكلاسيكية واكتشاف الكائنات (باستخدام Haar Cascades).
 - **NumPy:** لمعالجة البيانات الرقمية بكفاءة، خاصة مصفوفات الصور.
 - **python-decouple:** لإدارة متغيرات البيئة.
 - **gunicorn:** خادم WSGI لتشغيل تطبيق Django في بيئة الإنتاج.
- **قاعدة البيانات: (Database)**
 - **PostgreSQL:** قاعدة بيانات علائقية قوية وموثوقة، مستضافة على Render.com أو Supabase كقاعدة بيانات خارجية.
- **التخزين السحابي: (Cloud Storage)**
 - **Supabase Storage (S3 Compatible):** لتخزين الصور الأصلية والمعالجة بشكل آمن وقابل للتوسع.
- **النشر: (Deployment)**
 - **Render.com:** منصة سحابية لاستضافة تطبيق الويب وقاعدة البيانات.

3.2. تدفق البيانات

1. **المستخدم يرفع صورة:** يتم إرسال الصورة من الواجهة الأمامية (HTML Form) إلى `upload_view` في Django.
2. **معاينة الفلتر:** عند اختيار فلتر، يتم إرسال الصورة كـ (Data URL ونوع الفلتر ومعاملاته إلى `preview_filter_view` عبر طلب AJAX.
3. **معالجة الصورة: (Backend)** تستقبل دالة `apply_cv_filter` الصورة وتطبق الفلتر المطلوب باستخدام OpenCV.
4. **إعادة المعاينة:** يتم إرجاع الصورة المعالجة كـ (Data URL إلى الواجهة الأمامية لعرضها للمستخدم.
5. **حفظ الصورة:** عند تأكيد المستخدم، يتم حفظ الصورة الأصلية والمعالجة في Supabase Storage ، ويتم تسجيل بياناتها (المستخدم، الفلتر المطبق، روابط الصور) في قاعدة بيانات PostgreSQL.
6. **عرض الصور المحفوظة:** يستطيع المستخدم استعراض الصور المعالجة من قاعدة البيانات، ويتم جلبها من Supabase Storage لعرضها.

4. الميزات والوظائف

4.1. إدارة المستخدمين

يتيح النظام للمستخدمين إنشاء حسابات جديدة، تسجيل الدخول، تسجيل الخروج، وتحديث كلمات المرور الخاصة بهم.

4.2. رفع ومعاينة الصور

يمكن للمستخدمين رفع صورهم بسهولة. توفر الواجهة ميزة معاينة حية تسمح للمستخدم برؤية تأثير الفلتر المختار على الصورة قبل تطبيق التغييرات بشكل دائم وحفظها.

4.3. الفلاتر الكلاسيكية

هذه الفلاتر تعتمد على تقنيات معالجة الصور التقليدية:

- **الصورة الأصلية:** لعرض الصورة بدون أي تعديلات.

- **قص الصورة (Crop):** أداة تفاعلية لقص أجزاء محددة من الصورة.
- **تحويل إلى الرمادي (Grayscale):** تحويل الصورة إلى تدرجات اللون الرمادي.
- **قلب الألوان (Invert):** عكس قيم ألوان البكسلات في الصورة.
- **زيادة الحدة (Sharpen):** تحسين وضوح حواف الصورة.

4.4. فلتر الذكاء الاصطناعي (AI)

تستخدم هذه الفلاتر نماذج التعلم الآلي (Haar Cascades) لاكتشاف الكائنات وتطبيق التأثيرات الذكية:

- **طمس الوجوه (Blur Faces):** يكتشف الوجوه في الصورة ويطبق عليها تأثير التعتيم للحفاظ على الخصوصية.
- **التعرف على وجوه البشر (Detect Human Face):** يحدد ويرسم مستطيلات حول وجوه البشر المكتشفة.
- **التعرف على وجوه القطط (Detect Cat Face):** يحدد ويرسم مستطيلات حول وجوه القطط المكتشفة.
- **التعرف على العيون (Detect Eyes):** يحدد ويرسم مستطيلات حول العيون المكتشفة.
- **التعرف على الابتسامات (Detect Smile):** يحدد ويرسم مستطيلات حول الابتسامات المكتشفة.
- **التعرف على كامل الجسد (Detect Full Body):** يحدد ويرسم مستطيلات حول الأجسام الكاملة المكتشفة.
- **التعرف على الجزء العلوي من الجسد (Detect Upper Body):** يحدد ويرسم مستطيلات حول الأجزاء العلوية من الأجسام المكتشفة.
- **التعرف على لوحات السيارات (Detect License Plate):** يحدد ويرسم مستطيلات حول لوحات السيارات المكتشفة.

5. التنفيذ التقني

5.1 الواجهة الخلفية (Backend)

تم بناء الواجهة الخلفية باستخدام Django ، مما يوفر بنية MVC قوية. يتم استخدام OpenCV (cv2) لتنفيذ جميع عمليات معالجة الصور. تم تنظيم الفلاتر في دالة `apply_cv_filter` التي تستقبل الصورة ونوع الفلتر والمعاملات الخاصة به. تم استخدام Haar Cascades المتاحة في OpenCV لميزات الكشف عن الكائنات.

5.2 الواجهة الأمامية (Frontend)

تعتمد الواجهة الأمامية على HTML5 و Tailwind CSS لتصميم سريع الاستجابة وجذاب. يتم استخدام JavaScript لتوفير تجربة مستخدم ديناميكية، بما في ذلك:

- **أداة Cropper.js:** لتوفير وظيفة قص الصورة التفاعلية.
- **طلبات AJAX:** لإرسال الصور والفلاتر إلى الواجهة الخلفية للحصول على معاينات فورية دون إعادة تحميل الصفحة.
- **تبديل الفلاتر:** منطق JavaScript لتبديل عرض الفلاتر بين "كلاسيكية" و "ذكاء اصطناعي" بناءً على اختيار المستخدم.
- **الوضع الليلي/النهاري:** تبديل فئات CSS على `body` لتغيير سمة الواجهة، مع حفظ التفضيل في `localStorage`.
- **تعدد اللغات:** استخدام سمات `data-lang-en` و `data-lang-ar` مع JavaScript لتبديل محتوى النصوص بين العربية والإنجليزية.

5.3 قاعدة البيانات والتخزين

- **قاعدة البيانات:** تم استخدام PostgreSQL كقاعدة بيانات علائقية، مستضافة على Render.com. يتم تخزين بيانات المستخدمين، وسجلات الصور المعالجة، والفلاتر المطبقة.
- **التخزين السحابي:** يتم تخزين ملفات الصور الفعلية (الأصلية والمعالجة) في Supabase Storage ، الذي يوفر واجهة متوافقة مع S3 ، مما يضمن قابلية التوسع والأمان.

5.4 النشر (Deployment)

تم نشر التطبيق على منصة Render.com. تم استخدام ملف `render.yaml` لتحديد إعدادات النشر، بما في ذلك بيئة Python ، وأوامر البناء والتشغيل) باستخدام (Gunicorn ، ومتغيرات البيئة اللازمة) مثل مفاتيح (Supabase.

5.5 دعم تعدد اللغات والوضع الليلي/النهاري

تم تصميم الواجهة الأمامية لدعم تعدد اللغات (العربية والإنجليزية) من خلال سمات HTML المخصصة ومنطق JavaScript الذي يقوم بتبديل النصوص. كما تم تضمين وظيفة تبديل الوضع الليلي والنهاري، مما يوفر مرونة في العرض بناءً على تفضيلات المستخدم.

6. التحديات والحلول

واجه المشروع عدة تحديات رئيسية أثناء التطوير والنشر، أبرزها:

- **مشاكل الذاكرة (Out-of-Memory) على Render:** كانت مكتبة `rembg` (لإزالة الخلفيات) تستهلك كمية كبيرة من الذاكرة عند استيرادها، مما أدى إلى تعطل عمال `Gunicorn` على خطة `Render` المجانية.
 - **الحل:** تم إزالة مكتبة `rembg` بالكامل من المشروع، وتم حذف فلتر "إزالة الخلفية". هذا قلل بشكل كبير من استهلاك الذاكرة عند بدء تشغيل التطبيق والمعالجة، مما أدى إلى استقرار النشر.
- **خطأ `DisallowedHost` في `Django`:** كان التطبيق يرفض الاتصالات من دومين `Render` بسبب عدم تطابق في إعدادات `ALLOWED_HOSTS`.
 - **الحل:** تم التأكد من أن قيمة `ALLOWED_HOSTS` في `render.yaml` وفي متغيرات البيئة على `Render` تطابق تمامًا دومين التطبيق (على `Render` بدون `https://`).
- **عدم تطابق متغيرات البيئة:** (`decouple.UndefinedValueError`) كانت أسماء متغيرات البيئة في `render.yaml` لا تتطابق مع الأسماء التي تبحث عنها دالة `config()` في `settings.py`.
 - **الحل:** تم توحيد أسماء متغيرات البيئة الخاصة المتعلقة بـ (`Supabase S3`) لتكون متطابقة تمامًا في `settings.py`، `render.yaml`، وفي إعدادات `Render Dashboard`.
- **خطأ `No module named 'app'` في `Gunicorn`:** كان أمر بدء التشغيل في `Render` يشير إلى `app:app` بدلاً من المسار الصحيح لتطبيق `Django WSGI`.
 - **الحل:** تم تصحيح `startCommand` في `render.yaml` وفي إعدادات `Render Dashboard` إلى `gunicorn filter_studio.wsgi:application`.

7. العمل المستقبلي والتحسينات

هناك العديد من التحسينات والتوسعات المحتملة للمشروع:

- **إعادة دمج إزالة الخلفية:** إذا سمحت الموارد (مثل الترقية إلى خطة `Render` بذاكرة أكبر)، يمكن إعادة دمج ميزة إزالة الخلفية باستخدام `rembg` أو بدائل أخرى.
- **فلتر `AI` إضافية:** استكشاف دمج المزيد من فلاتر الذكاء الاصطناعي المتقدمة (مثل تحويل الأنماط، تحسين جودة الصورة، أو توليد الصور).
- **أدوات تعديل يدوية:** إضافة أدوات تعديل يدوية مثل تعديل السطوع، التباين، التشبع، وتوازن الألوان.
- **معرض صور عام:** السماح للمستخدمين بمشاركة صورهم المعالجة في معرض عام.
- **نظام اشتراكات:** تقديم خطط اشتراك لميزات متقدمة أو سعة تخزين أكبر.
- **تحسين الأداء:** استكشاف تقنيات إضافية لتحسين سرعة معالجة الصور، مثل استخدام الحوسبة المتوازية أو `GPUs` إذا كانت متاحة في بيئة النشر.
- **اختبارات الوحدة والتكامل:** إضافة اختبارات شاملة لضمان جودة الكود واستقراره.
- **تحسين تجربة المستخدم:** إضافة رسوم متحركة أكثر سلاسة، إشعارات أفضل، وتحسينات في قابلية الوصول.
- **نظام التعرف على الوجوه الآمن** (`Face Recognition (FaceNet)` - تشفير البيانات (`PyCrypto`) | كشف هويات مفترضة مع حماية البيانات
- **كشف الأجسام المشبوهة** - `YOLOv8` (كشف أسلحة/مركبات) `OpenCV Tracking` - تحليل تهديدات أمنية في الصور
- **معالجة الصور الجوية** (`GIS processing (Rasterio)` - تحسين دقة الصور (`SRGAN`) | تحليل صور الأقمار الصناعية والطائرات المسيرة

8. الخاتمة

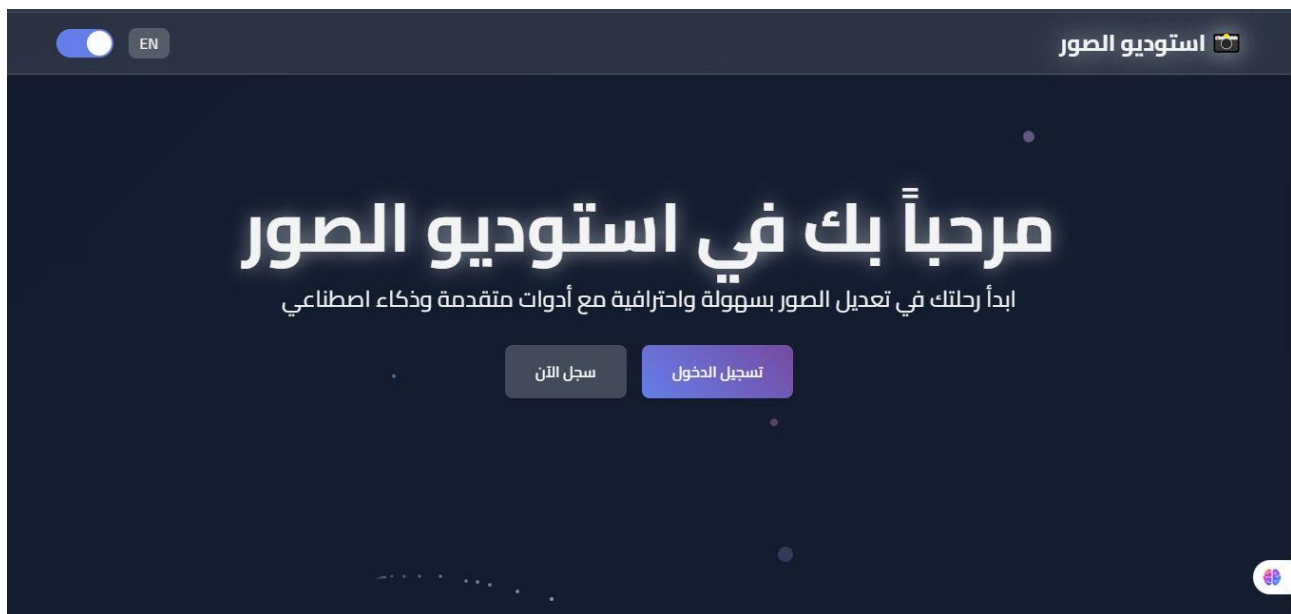
يمثل مشروع "استوديو الصور" تطبيق ويب وظيفيًا لمعالجة الصور، يقدم مجموعة من الفلاتر الكلاسيكية والذكاء الاصطناعي في واجهة سهلة الاستخدام. على الرغم من التحديات التي واجهت عملية النشر، تم التغلب عليها بنجاح، مما أدى إلى تطبيق مستقر وقابل للتشغيل على Render.com. يوفر المشروع أساسًا قويًا لمزيد من التطوير والتحسين في مجال معالجة الصور الرقمية.

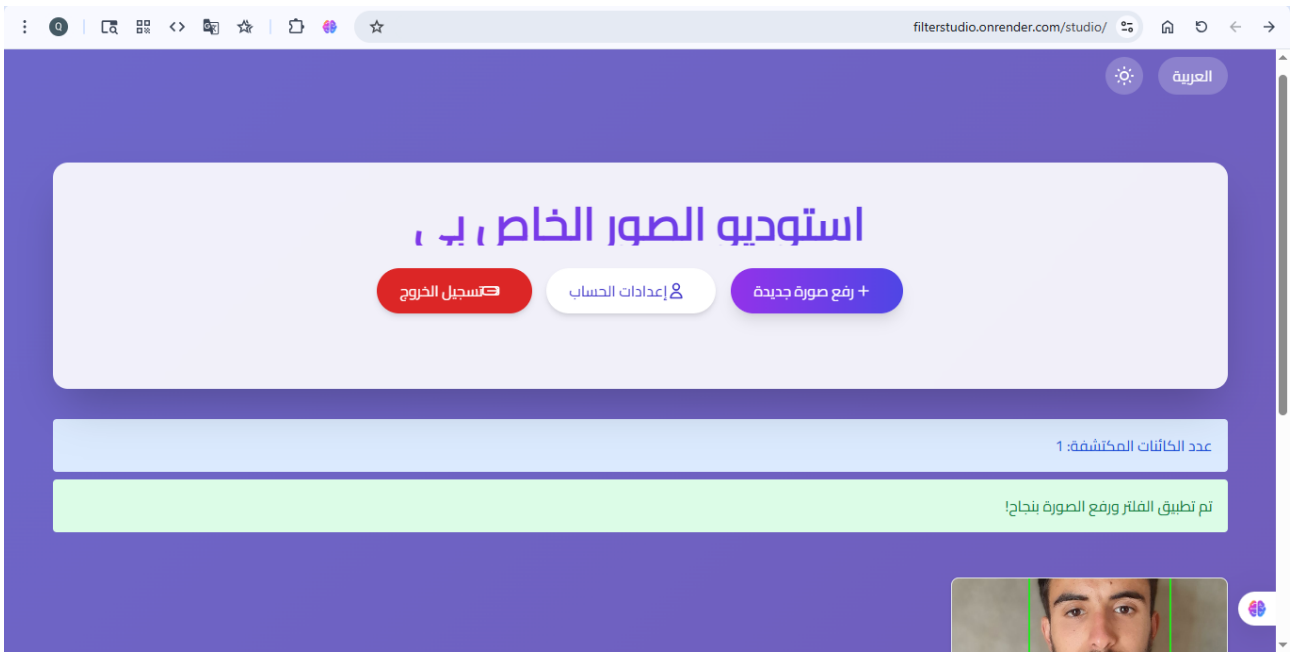
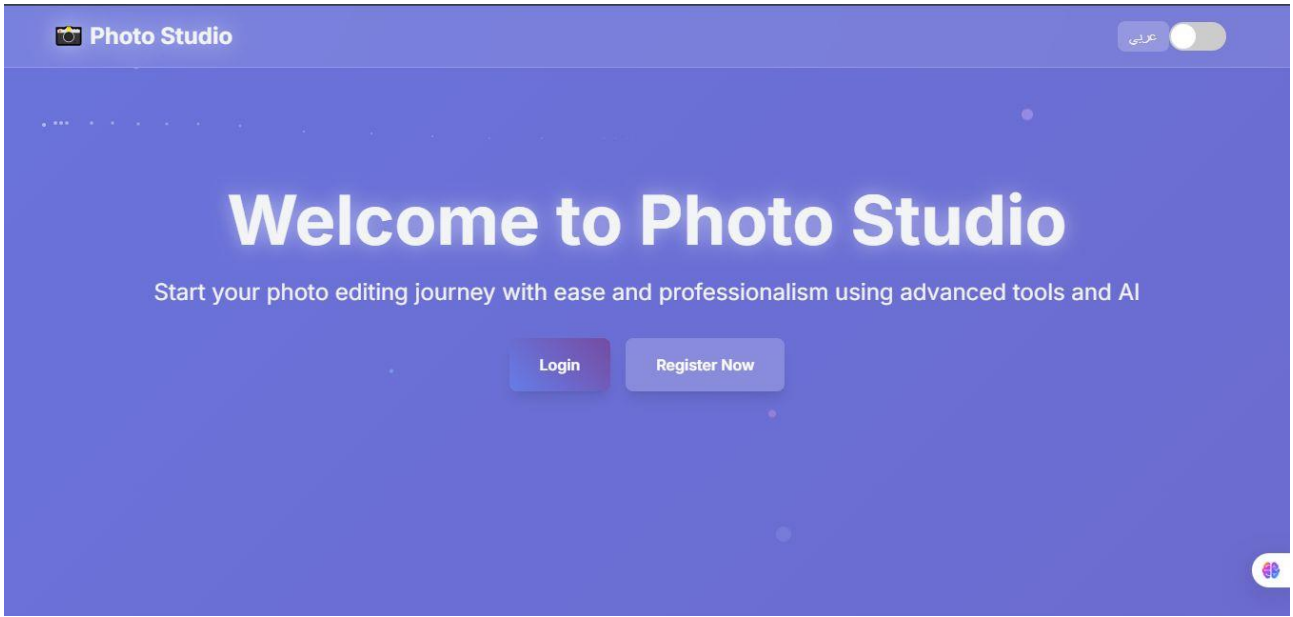
9. المراجع

- وثائق Django الرسمية: <https://docs.djangoproject.com/>
- وثائق OpenCV-Python: <https://docs.opencv.org/>
- وثائق Tailwind CSS: <https://tailwindcss.com/docs>
- وثائق Render.com: <https://render.com/docs>
- وثائق Supabase: <https://supabase.com/docs>
- وثائق Cropper.js: <https://fengyuanchen.github.io/cropperjs/>

رابط الموقع : <https://filterstudio.onrender.com/>

- 10. الملاحق
- الملحق أ: لقطات شاشة من التطبيق





filterstudio.onrender.com/image/22/

☆

تفاصيل الصورة

معلومات الصورة

اسم المستخدم: test@22


نوع الفلتر: التعرف على وجوه البشر

تاريخ المعالجة: 15:06 2025-07-20

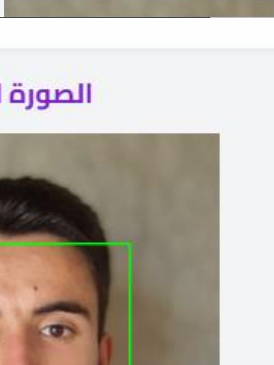
تحميل الصورة المعدلة

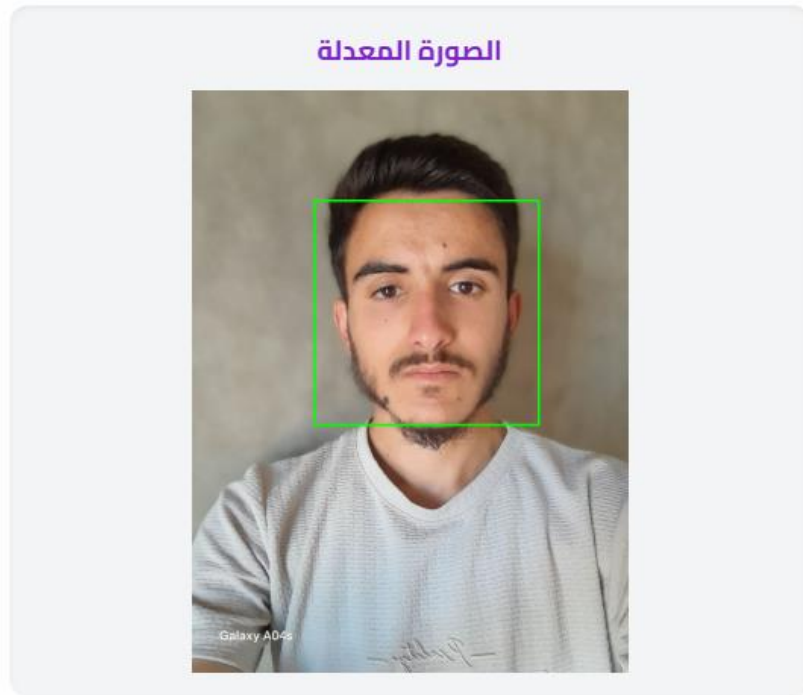
حذف الصورة

الصورة الأصلية



الصورة المعدلة





رفع وتعديل صورة

تم تعميم 1 وجه.

1. اختر صورة

hamid7.jpg

اختيار ملف



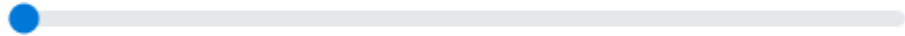
Galaxy A04s

2. اختر الفلتر



طمس الوجوه (تفاعلي)

درجة التعقيم: 25

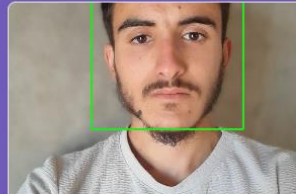


تطبيق ورفع

العودة إلى الاستوديو

تم تعميم 1 وجه.

تم تطبيق الفلتر ورفع الصورة بنجاح!



الفلتر: التعرف على وجوه البشر

تاريخ المعالجة: 15:06 20-07-2025

حذف

التفاصيل



الفلتر: طمس الوجوه (تفاعلي)

تاريخ المعالجة: 15:12 20-07-2025

حذف

التفاصيل