نظام الكشف عن الالتهاب الرئوي من أشعة الصدر

Chest X-ray Pneumonia Detection System

شرح تقني متعمّق: الـ AI pipeline كامل + الـ web application

شرح بالمصري للمشروع كله من أول الـ data لحد ما الـ model يشتغل في الموقع، بالتفاصيل والأرقام والصور.

Team 19 - School of CSAI, Zewail City of Science and Technology - Supervisor: Prof. Dr. Khaled Mostafa

June 2026 -

الورقة دي مكتوبة عشان أي حد يقراها يقدر يفهم المشروع كله ويشرحه بثقة، حتى لو أول مرة يشوفه. فيها قسمين: القسم A بيشرح الـ AI pipeline (الـ data، الـ preprocessing، الـ models، التدريب، التقييم، والـ XAI)، والقسم B بيشرح الـ web application (الـ frontend والـ backend والـ database والـ security ورحلة الطلب من الأول للآخر). كل قسم بينتهي ببنك أسئلة متوقّعة من لجنة المناقشة بإجاباتها.

Table of Contents - الفهرس

Table of Contents - الفهرس 2

AI Pipeline الـ - A القسم 3

A1. الـ Dataset: RSNA Pneumonia Detection Challenge 3

A2. من صورة خام لـ model input (الـ Preprocessing) 3

A3. تقسيم الـ data صح + معالجة عدم توازن الـ classes (imbalance) 3

A4. مهمتين: classification و detection 4

A5. الـ Models بالتفصيل + عدد الـ parameters 4

A6. تدريب الـ classifiers (كل الإعدادات) 5

A7. تدريب الـ detectors 5

A8. الـ Nature-Inspired Optimization: PSO و GWO و SA 5

A9. مقاييس التقييم (Metrics) بشرح بسيط 6

A10. النتائج، وليه طلعت كده، وإزاي نحسّنها 6

A11. الـ Explainability (XAI): نوري الدكتور 'ليه' 8

A12. القسم A - أسئلة متوقّعة من اللجنة 10

Web Application Pipeline الـ - B القسم 12

B1. الـ Frontend (Angular 21) 12

B2. الـ Backend (FastAPI) 13

B3. الـ Inference Service خطوة بخطوة 13

B4. الـ Security 14

B5. الـ Database (MongoDB) 14

B6. الـ API 14

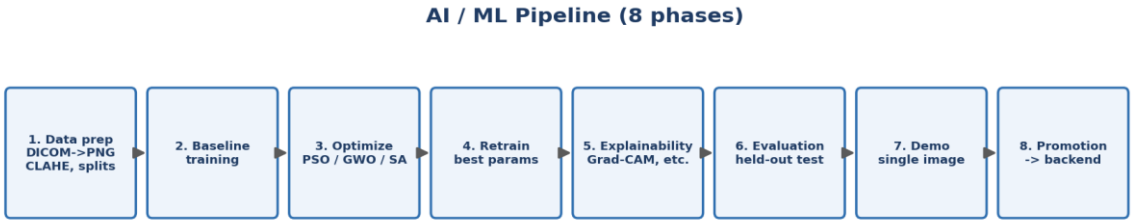
B7. رحلة طلب كاملة من الأول للآخر 15

B8. الـ Deployment والإعداد 15

B9. القسم B - أسئلة متوقّعة من اللجنة 16

القسم A - الـ AI Pipeline

باختصار في جملة: بناخد صورة أشعة صدر (chest X-ray)، بنعدّيها على deep-learning model مدرّب، وبيرجّعلنا يا إما احتمال إن الرئة فيها التهاب رئوي (دي مهمة الـ classification)، يا إما box حوالين مكان الالتهاب مع نسبة ثقة (دي مهمة الـ detection). كل اللي تحت بيشرح إزاي بنوصل من صورة طبية خام للإجابة دي، وليه بتطلع كده.



شكل A1. مراحل الـ AI pipeline الثمانية (8 -> 1 phases).

A1. الـ Dataset: RSNA Pneumonia Detection Challenge

استخدمنا dataset بتاع تحدي RSNA Pneumonia Detection اللي نشرته الـ Radiological Society of North America. فيه حوالي 26,684 صورة أشعة صدر أمامية (frontal chest X-rays)، كل واحدة اتعلّمت (labeled) من radiologists. أهم نقطة إنه بيدّينا نوعين من الـ label: labels على مستوى الصورة (فيه التهاب ولا لأ) بنستخدمه في الـ classification، وbounding boxes (المستطيلات اللي بتحدد مكان الـ opacity اللي بيشبه الالتهاب) بنستخدمها في الـ detection. اخترناه لأنه كبير، متعلّم بإيد متخصصين، ومستخدَم كـ benchmark مشهور (يعني أرقامنا تقدر تتقارن بغيرنا)، وبيدعم المهمتين مع بعض.

ليه ده مهم؟ جودة أي model مسقوفة بجودة الـ data بتاعته. وبما إن RSNA متعلّم من خبراء، النتائج بتاعتنا بتعكس الـ model نفسه، مش noise إحنا اللي دخّلناه.

A2. من صورة خام لـ model input (الـ Preprocessing)

الأشعة الطبية مبتجيش صور عادية، بتيجي على هيئة DICOM files، وده format طبي بيخزّن الـ pixels مع بيانات المريض. الـ preprocessing بتاعنا بيحوّل كل DICOM لـ PNG نضيفة الـ model يقدر يقراها، بالخطوات دي:

- قراءة الـ DICOM بمكتبة pydicom عشان نطلّع مصفوفة الـ grayscale pixels الخام.
- **Robust intensity normalization:** سطوع الأشعة بيختلف من جهاز للتاني، فبنقص (clip) كل صورة عند الـ 2nd-98th percentiles وبنرجّعها لمدى 0-255. ده بيشيل الـ outliers ويخلّي الصور قابلة للمقارنة.
- **CLAHE contrast enhancement:** الـ Contrast Limited Adaptive Histogram Equalization (clip limit = 2.0، tiles = 8x8) بيزوّد الـ contrast محليًا عشان الـ opacities الخفيفة في الرئة تبان. بيعمل equalize في مربعات صغيّرة مش على الصورة كلها، وده مثالي للأشعة.
- Resize لمقاس ثابت (224x224 للـ classifiers، و640x640 للـ detectors) مع حفظ الأبعاد الأصلية عشان نقدر نرجّع إحداثيات الـ boxes صح.

أهم درس اتعلمناه هنا (وغالبًا سؤال امتحان): في الأول كانت النتايج عالية بشكل مريب. السبب كان data leak: الـ contrast enhancement كان متطبّق بشكل مختلف على صور الالتهاب وصور الطبيعي، فالـ model بقى بيغش وبيكتشف الـ preprocessing مش المرض. صلّحناها بإننا نطبّق نفس الـ CLAHE بالظبط على كل الصور بغض النظر عن الـ label. الأرقام الأقل اللي بنعرضها دلوقتي هي الحقيقية. الدرس: ثِق في البروتوكول مش في الرقم.

A3. تقسيم الـ data صح + معالجة عدم توازن الـ classes (imbalance)

قسّمنا الـ data لـ train / validation / test على مستوى المريض (patient-level) مش على مستوى الصورة، وبـ seed ثابت (42) عشان النتائج تتكرر. المريض الواحد ممكن يكون عنده أكتر من صورة، ولو صور نفس المريض اتوزّعت بين الـ train والـ test، الـ model ممكن يحفظ المريض ده وياخد درجة أعلى من الحقيقة. التقسيم بالمريض بيضمن إن الـ test set فيه ناس الـ model عمره ما

شافها. والتقسيم كمان stratified يعني نسبة الالتهاب للطبيعي محفوظة في كل جزء. الـ test set المعزول = 20% من المرضى: 4,135 صورة طبيعية و1,202 صورة التهاب، ومبنلمسهوش خالص أثناء التدريب أو اختيار الـ model.

عدم التوازن (class imbalance): الالتهاب هو الـ class الأقل (minority)، فلو سبنا الـ model زي ما هو ممكن ياخد accuracy عالية بإنه يقول 'طبيعي' على طول. عالجنا ده بـ class-weighted CrossEntropy: بنّدّي وزن أكبر للـ class النادر بنسبة عكسية لتكراره (inverse class frequency)، فالـ model بيتعاقب أكتر لما يفوّت حالة التهاب. وكمان بنقيّم بـ AUC و recall والـ confusion matrix مش بالـ accuracy لوحدها. وعلى مستوى الصور بنستخدم augmentation وقت التدريب بس (h-flip، color jitter، rotation) عشان نقلّل الـ overfitting.

A4. مهمتين: classification و detection

الـ classification بيجاوب على سؤال 'هل فيه التهاب في الصورة دي؟' باحتمال. والـ detection بيجاوب 'فين بالظبط؟' برسم box أو أكتر، كل واحد معاه نسبة ثقة. عملنا الاتنين لأن الدكتور عايز screening سريع (نعم/لأ) وكمان مؤشّر بصري للمكان المشكوك فيه. الـ model اللي بننشره كـ default هو detector (Faster R-CNN) لأن تحديد المكان أنفع إكلينيكيًا.

A5. الـ Models بالتفصيل + عدد الـ parameters

كل الـ models الخمسة بتستخدم transfer learning: بدل ما تتعلّم الرؤية من الصفر، بتبدأ من weights مدرّبة على ImageNet (dataset ضخم لصور طبيعية) وبعدين بنعمل fine-tune على أشعة الصدر. ده standard في الـ medical imaging لأن الـ datasets الطبية صغيّرة، والـ low-level features (الحواف والـ textures) بتنتقل كويس.

الفكرة المميّزة	عدد الـ Parameters	النوع	الـ Model
residual connections (شورت كت بيخلّي الشبكة العميقة تتدرّب)	23.5M	classifier	ResNet50
كل layer متوصّلة بكل اللي بعدها، فالـ features بتتعاد استخدامها (backbone بتاع CheXNet)	7.0M	classifier	DenseNet121
compound scaling - أصغر classifier وأكفأ واحد عندنا	4.0M	classifier	EfficientNet-B0
two-stage: RPN يقترح boxes وبعدين head يصنّف ويظبط - أعلى recall (المنشور)	41.3M	detector	Faster R-CNN
one-stage anchor-free - أسرع وأخف بس أقل recall	3.0M	detector	YOLOv8n

الـ classifiers الثلاثة

- **ResNet50:** شبكة convolutional من 50 layer، فكرتها الأساسية الـ residual connection (شورت كت بيخلّي الإشارة تتخطّى layers)، وده حلّ مشكلة إن الشبكات العميقة جدًا صعب تتدرّب.
- **DenseNet121:** كل layer متوصّلة بكل اللي بعدها فبيحصل feature reuse على طول الشبكة. كفؤة في الـ parameters وكانت الـ backbone بتاع CheXNet، الموديل الشهير اللي وصل لمستوى الـ radiologist.
- **EfficientNet-B0:** بيستخدم compound scaling عشان يوازن بين العمق والعرض ودقة الـ input. هو أحسن classifier عندنا وكمان أصغرهم (4M parameters)، فهو الأكفأ.

الـ detectors الاتنين

- **Faster R-CNN (two-stage):** الأول Region Proposal Network بيقترح boxes مرشّحة، وبعدين head تاني بيصنّف كل box ويظبط إحداثياته. بيستخدم ResNet50 backbone مع Feature Pyramid Network عشان يشوف الأحجام المختلفة. دقيق خصوصًا في الـ recall، وعشان كده بننشره.
- **YOLOv8 (one-stage):** بيتنبأ بكل الـ boxes في forward pass واحدة بـ head من غير anchors. أسرع وأصغر بكتير، بس في اختباراتنا فوّت حالات التهاب أكتر (recall أقل) من Faster R-CNN.

A6. تدريب الـ classifiers (كل الإعدادات)

الإعداد	القيمة	السبب
Optimizer	Adam	خيار افتراضي قوي ومتكيّف للـ fine-tuning
Learning rate	1e-4	صغير عشان مايخربش الـ pretrained weights
Loss	Cross-entropy موزون بالـ class	بيدّي وزن للـ class النادر (الالتهاب) فمايتجاهلوش
LR scheduler	ReduceLROnPlateau (على val AUC، x0.5)	بيقلّل الـ LR لما الـ validation يبطّل يتحسّن
Mixed precision (AMP)	On على الـ GPU	تدريب أسرع وذاكرة أقل
Early stopping	patience = 4 على val AUC	نوقف قبل الـ overfitting ونرجّع أحسن weights
Augmentation (train بس)	h-flip، rotate 10، color jitter	بيعلّم الـ model الثبات ويقلّل الـ overfitting
Normalization	ImageNet mean/std	يطابق الـ pretrained backbone
Image size	224 224 x	المقاس القياسي للـ backbones دي

نقطتين مهمين. **Class weighting:** الالتهاب هو الأقلية، فبنوزن الـ loss بنسبة عكسية لتكرار الـ class، من غيرها الـ model ممكن ياخد accuracy عالية بإنه يقول 'طبيعي' دايمًا. **Best-checkpoint restore:** بنحتفظ بالـ weights من الـ epoch اللي عندها أحسن validation AUC، مش آخر epoch، عشان منشحنش model عمل overfit.

A7. تدريب الـ detectors

Faster R-CNN بيتدرّب على الـ TorchVision training loop: optimizer = SGD (momentum 0.9)، learning-rate schedule = linear warmup بعدها cosine decay، تجميد الـ backbone في أول epoch عشان الـ heads الجديدة تستقر قبل ما نعمل fine-tune للشبكة كلها، AMP، early stopping على mAP@0.5، وbest-checkpoint restore. بيتحسّن على الـ Faster R-CNN multi-task loss (objectness + box للـ RPN، وclassification + box-regression للـ head). وقت الـ inference بنطبّق Non-Maximum Suppression (NMS) عشان نشيل الـ boxes المكررة المتداخلة (مشروح في القسم B). أما YOLOv8 فبيتدرّب من خلال الـ Ultralytics pipeline بالـ augmentation الجاهز بتاعه.

A8. الـ Nature-Inspired Optimization: PSO و GWO و SA

عشان ندوّر على إعدادات (hyperparameters) أحسن، طبّقنا ثلاث خوارزميات مستوحاة من الطبيعة. كلهم طرق للبحث في فضاء من الإعدادات من غير ما نجرّب كل الاحتمالات:

- **Particle Swarm Optimization (PSO):** سرب من الحلول المرشّحة 'بيطير' في فضاء البحث، كل واحد بينجذب لأحسن نتيجة ليه ولأحسن نتيجة للسرب كله، زي العصافير اللي بتلمّ على الأكل.
- **Grey Wolf Optimizer (GWO):** بيقلّد هرم قطيع الدياب؛ أحسن ثلاث حلول (alpha و beta و delta) بيقودوا باقي القطيع ناحية المناطق الواعدة.
- **Simulated Annealing (SA):** مستوحى من تبريد المعدن؛ بيقبل أحيانًا حلول أسوأ في الأول (حرارة عالية) عشان يهرب من الـ local traps، وبعدين بيستقر مع ما بيبرد.

النتيجة الصادقة (سؤال متوقّع): في النسخة الأولى قيّمنا كل candidate بـ proxy رخيص (epoch واحدة على جزء من الـ data وبـ budget صغير: population=3, iterations=2) عشان البحث يخلّص في session واحدة على Kaggle. الـ proxy ده طلع noisy جدًا: الإعدادات اللي اختارها مكانتش بتكسب بعد التدريب الكامل، وفي حالات خلّت النتيجة أسوأ من الـ baseline (مثلًا Faster R-CNN mAP@0.5 نزلت من 0.381 لـ 0.175، وEfficientNet AUC من 0.886 لـ 0.874). الدرس إن proxy search خفيف مش بالضرورة بيتفوّق على baseline متظبوط، وده finding حقيقي ومحترم مش فشل.

الحل: إعادة بحث أقوى (Stronger re-run)

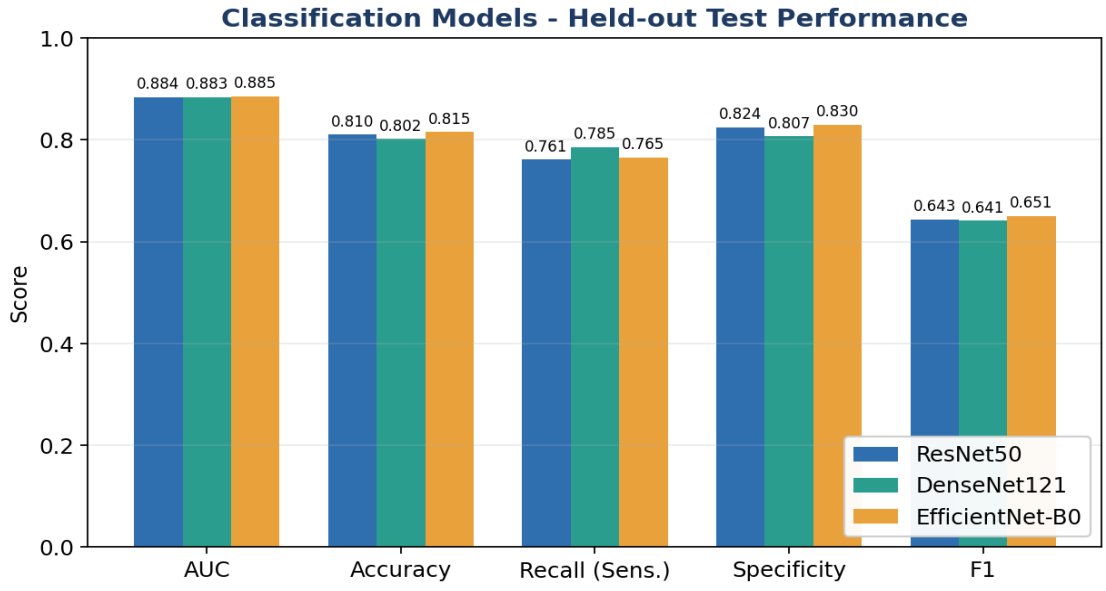
ضفنا في كل notebook قسم 'Stronger nature-inspired search + retrain' بيعيد الـ Phase 4 بـ budget أكبر **ودقّة proxy أعلى** (epochs أكتر و data أكتر لكل candidate، عشان الإعدادات اللي بتتختار تنفع فعلًا في التدريب الكامل)، وبعدين

بيعمل retrain كامل. القسم ده بيقرأ الـ baseline المحفوظ من results/<model>_report.json عشان يقارن before/after، فمش محتاج يعيد تدريب الـ Phase 3 خالص، وبيستبدل الـ checkpoint المنشور بس لو النسخة الجديدة كسبت الـ baseline. الأرقام النهائية للجزء ده بتتحدّث بعد ما نشغّله على Kaggle.

A9. مقاييس التقييم (Metrics) بشرح بسيط

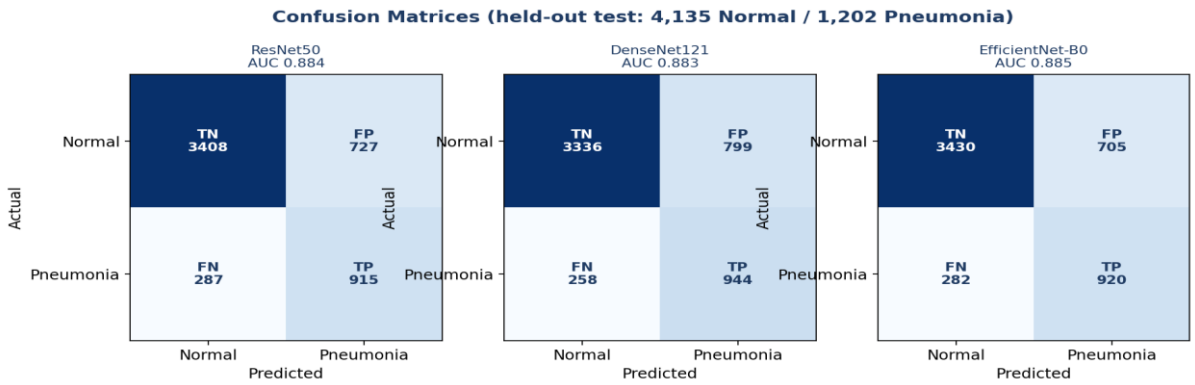
- **Accuracy:** نسبة التنبؤات الصح من الكل. مضلِّلة على الـ data غير المتوازنة، فمابنعتمدش عليها لوحدها.
- **Recall / Sensitivity:** من المرضى الحقيقيين، كام واحد مسكناه. ده أهم مقياس هنا، لأن حالة التهاب فايتة ممكن تكون خطر على الحياة.
- **Specificity:** من الأصحاء الحقيقيين، كام واحد طلّعناه سليم صح.
- **Precision:** من اللي قلنا عليهم مرضى، كام واحد فعلًا عنده التهاب.
- **F1:** التوازن (harmonic mean) بين الـ precision والـ recall.
- **AUC:** المساحة تحت الـ ROC curve؛ احتمال إن الـ model يرتّب مريض عشوائي أعلى من سليم عشوائي. 0.5 عشوائي و1.0 مثالي. مش معتمد على threshold، فهو المقياس الرئيسي عندنا للـ classification.
- **Confusion matrix:** جدول الـ true/false positives و negatives اللي بنحسب منه كل اللي فوق.
- **mAP و IoU و mAP@0.5:** للـ detection. الـ IoU (Intersection over Union) بيقيس تداخل الـ boxes؛ الـ box بيتحسب صح لو الـ IoU بتاعه مع الحقيقة عدّى threshold (مثلًا 0.5). الـ mAP متوسط الـ precision عبر مستويات الـ recall؛ mAP@0.5 بيستخدم threshold = 0.5، و mAP@[.5:.95] بيتوسّط على thresholds أصعب.

A10. النتائج، وليه طلعت كده، وإزاي نحسّنها



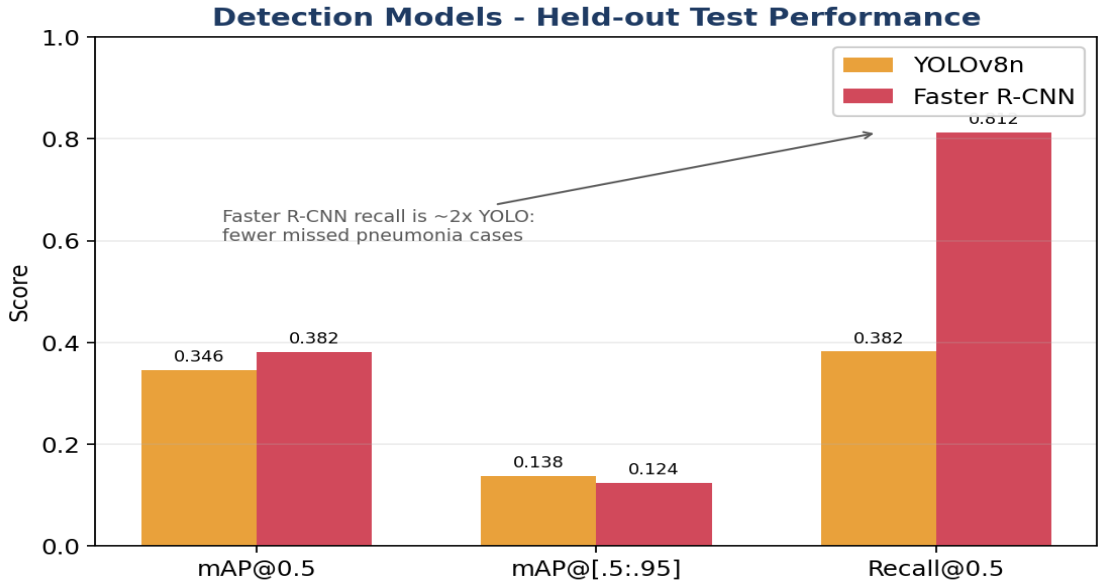
شكل A2. مقارنة أداء الـ classifiers الثلاثة على الـ test set.

Params	F1	Recall	Accuracy	AUC	Model (classifier) الـ
4.0M	0.651	0.765	0.815	0.886	EfficientNet-B0 (الأحسن، منشور)
7.0M	0.641	0.785	0.802	0.883	DenseNet121 (منشور)
23.5M	0.643	0.761	0.810	0.884	ResNet50



شكل A3. الـ confusion matrices للـ classifiers الثلاثة على الـ test set.

Params	Recall	mAP@[.5:.95]	mAP@0.5	الـ Model (detector)
41.3M	0.812	0.124	0.381	Faster R-CNN (المنشور)
3.0M	0.382	0.138	0.346	YOLOv8n



شكل A4. مقارنة الـ detectors (YOLO مقابل Faster R-CNN).

ليه الأرقام دي؟ AUC حوالي 0.88 هو اللي بيوصله أي classifier صادق على نوع الـ data ده؛ للمقارنة، CheXNet المشهور سجّل 0.768 على dataset تاني. الـ classifiers بيحافظوا على specificity عالية (حوالي 0.83) ومع كده بيمسكوا حوالي 76-79% من حالات الالتهاب. في الـ detection، الـ detectors بيوصلوا لـ mAP@0.5 متقارب (حوالي 0.35-0.38، وده بالظبط المدى 0.32-0.39 المذكور في الأبحاث)، بس Faster R-CNN بيمسك recall = 0.812 من مناطق الالتهاب مقابل 0.382 لـ YOLO.

ليه Faster R-CNN هو الأحسن (والمنشور)؟

الخطأ الأغلى في الـ screening هو إننا نفوّت حالة (false negative). Faster R-CNN عنده أعلى recall بفارق كبير (0.812 مقابل 0.382 لـ YOLO)، يعني بيفوّت أقل حالات بكتير، وكمان أعلى mAP@0.5 بين الـ detectors. عشان كده بننشره كـ default رغم إنه أكبر (41.3M parameters) وأبطأ. الـ trade-off مقصود: في الطب، مسك الحالة أهم من السرعة.

إزاي نطلّع نتايج أحسن؟ ندرّب على data أكتر وأكثر تنوّعًا (مستشفيات وأجهزة مختلفة)؛ نرفع دقة الـ input عشان الـ opacities الخفيفة تفضل بانة؛ نعمل ensemble لأكتر من model؛ نظبط الـ decision threshold عشان نبادل specificity بـ recall حسب احتياج العيادة؛ نضيف augmentation أقوى وواقعي طبيًا؛ نعمل pretrain على corpus كبير لأشعة الصدر قبل الـ fine-tune؛ وللـ detection، ندرّب أطول بالـ objective الكامل (مش proxy). وكمان external validation على dataset تاني هيخلّي النتايج أكتر مصداقية.

A11. الـ Explainability (XAI): نوري الدكتور "ليه"

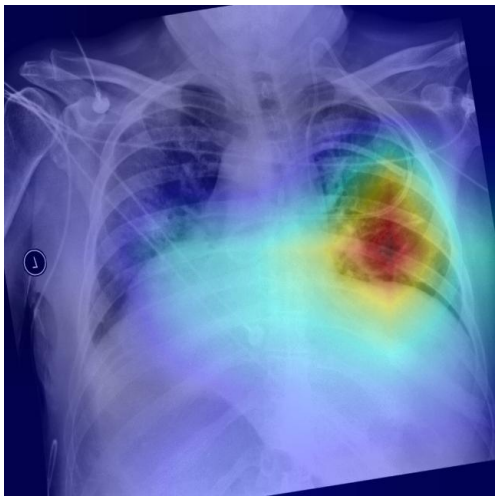
كل تنبؤ بنعرضه مع heatmap عشان الدكتور يتأكد إن الـ model بصّ على الرئة، مش على نص أو artifact. ده بيبني الثقة وبيخلّي القرار قابل للمراجعة. طبّقنا أكتر من طريقة، وكلها بتشتغل live مع كل طلب، وكلها best-effort (لو طريقة فشلت بتتخطّى من غير ما تكسر التنبؤ).

للـ classifiers

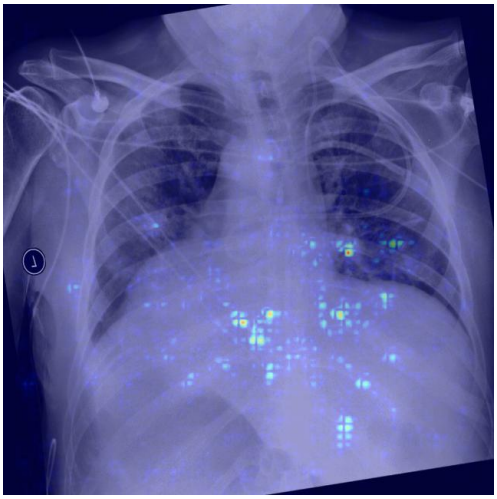
- **Grad-CAM:** بيستخدم الـ gradients عند آخر convolutional layer عشان يطلّع heatmap للمناطق اللي زوّدت احتمال الالتهاب. أكتر طريقة شائعة ومباشرة.
- **Integrated Gradients:** بيجمع الـ gradient على طول مسار من صورة سودا (baseline) للصورة الحقيقية، فبيدّي attribution لكل pixel.
- **GradientSHAP:** تقدير على طريقة SHAP بياخد متوسط gradient * (input - baseline) على عينات فيها noise.
- **Score-CAM:** من غير gradients؛ بيقنّع الصورة بكل activation channel ويوزنه بالـ pneumonia score الناتج، فبيدّي class-discriminative map.

مثال حقيقي - حالة Pneumonia: كل الصور دي نواتج فعلية من الـ web application على نفس صورة الأشعة. الـ classifier (EfficientNet-B0) قال **PNEUMONIA باحتمال 75%**. تحت كل صورة مكتوب التنبؤ، والـ heatmaps بتركّز على الرئة اليسرى (مكان الـ opacity) مش على حرف أو حافة.

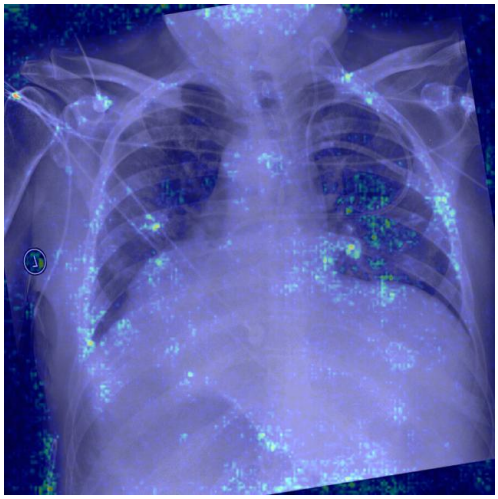
PNEUMONIA (75%) :التنبؤ - EfficientNet-B0



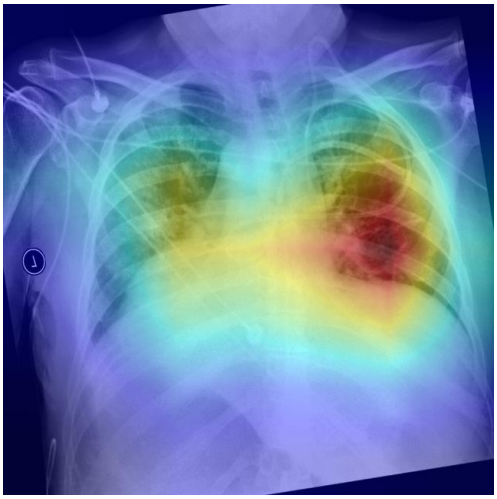
Grad-CAM - بيركّز على الرئة المصابة



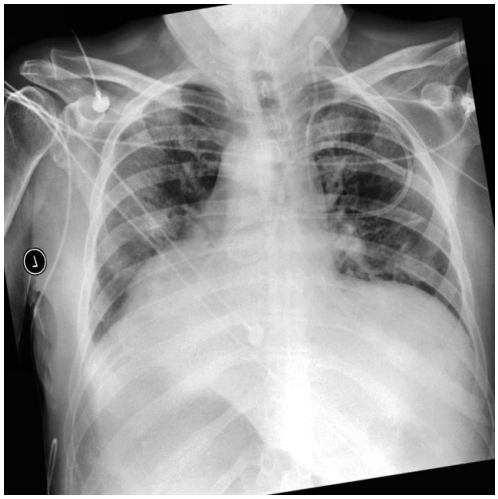
Integrated Gradients - PNEUMONIA



GradientSHAP - PNEUMONIA



Score-CAM (من غير gradients) - PNEUMONIA

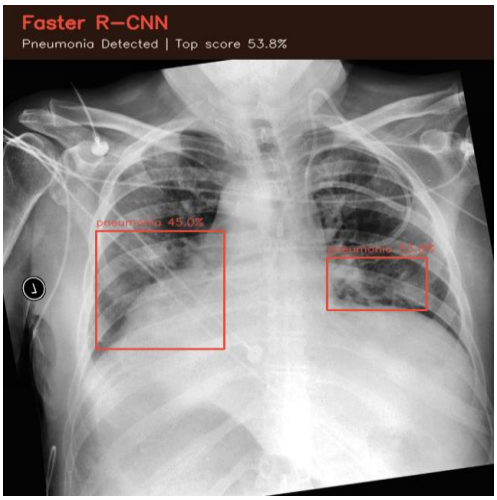


الصورة الأصلية (input X-ray)

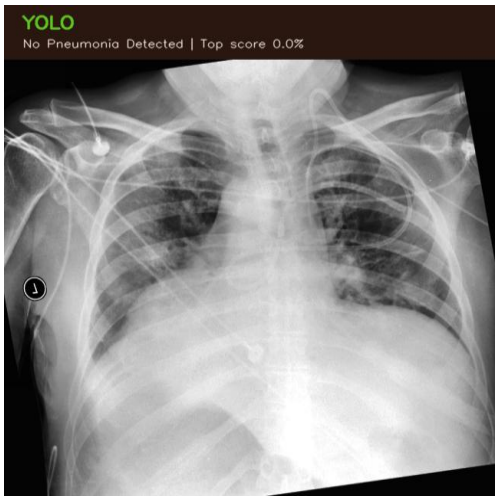
للـ detectors

- **Eigen-CAM:** الـ saliency القياسي بالـ gradients مبيتطبّقش بسهولة على الـ detectors، فبنستخدم Eigen-CAM اللي بيحلّل الـ activations بتاعة الـ backbone (المكوّن الرئيسي - principal component) عشان يبيّن الشبكة بتركّز فين.
- **Occlusion sensitivity:** بنغطّي أجزاء من الصورة بمربع ونشوف ثقة الـ detection بتنزل قد إيه؛ كل ما النزول أكبر، كل ما المنطقة دي أهم.

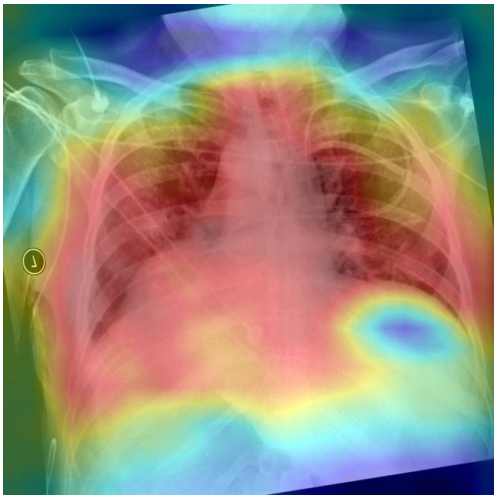
نفس الصورة بالـ detectors: Faster R-CNN لقى الالتهاب ورسم box (نسبة 54%، ومكتوب **Pneumonia Detected** فوق الصورة)، لكن YOLO فوّت الحالة (no detection) - وده بالظبط سبب نشرنا لـ Faster R-CNN: الـ recall الأعلى. والـ Eigen-CAM والـ Occlusion بيوضّحوا منطقة تركيز الـ detector.



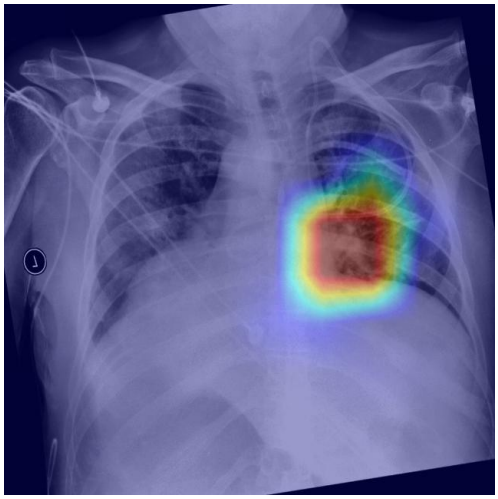
Pneumonia Detected (box) :التنبؤ - Faster R-CNN



فوّت الحالة (no detection) - YOLOv8



Eigen-CAM (Faster R-CNN backbone)



Occlusion sensitivity (Faster R-CNN)

للمقارنة - حالة Normal: على أشعة صدر سليمة، نفس الـ classifier قال **Normal باحتمال 99%** (احتمال الالتهاب 0.8% بس)، فالنظام بيفرّق بوضوح بين الحالتين.

أشعة صدر سليمة (input)

التنبؤ: (99%) Normal - EfficientNet-B0

ملاحظة: كل الصور دي نواتج حقيقية من الـ web application الشغّال. الـ heatmaps في حالة الالتهاب بتركّز على الرئة المصابة، وده بيأكد إن الـ models بتتعلّم إشارات طبية حقيقية مش artifacts.

A12. القسم A - أسئلة متوقّعة من اللجنة

س: ليه نتايجكوا الأولى كانت شبه مثالية وعملتوا إيه؟

ج: كانت متضخّمة بسبب data leak في الـ preprocessing: الـ contrast enhancement كان متطبّق بشكل مختلف على الـ class اتنين، فالـ model كان بيكتشف الـ preprocessing مش المرض. صلّحناها بتطبيق نفس الـ CLAHE على كل الصور وبنعرض الأرقام الصادقة الأقل.

س: ليه التقسيم بالمريض مش بالصورة؟

ج: لأن المريض الواحد ممكن يكون عنده أكتر من صورة. التقسيم بالصورة ممكن يحط نفس المريض في الـ train والـ test، فالـ model يحفظه وياخد درجة أعلى من الحقيقة. التقسيم بالمريض بيضمن إن الـ test مش متشاف.

س: ليه الـ recall هو أهم مقياس عندكوا؟

ج: لأن في screening الالتهاب، الـ false negative (حالة فايتة) ممكن يكون خطر على الحياة، أما الـ false positive بيأدي لمراجعة تانية بس. فبنختار ونظبط الـ models عشان نقلّل الحالات الفايتة.

س: ليه بتنشروا Faster R-CNN مش YOLO الأسرع؟

ج: الاتنين بيوصلوا mAP@0.5 متقارب، بس Faster R-CNN بيمسك recall = 0.812 مقابل 0.382 لـ YOLO. بنقبل الـ model الأكبر والأبطأ لأن مسك الحالات أهم من السرعة هنا.

س: عالجتوا عدم التوازن (imbalance) إزاي؟

ج: وزّنا الـ loss بنسبة عكسية لتكرار الـ class عشان الـ class النادر مايتجاهلش، وبنقيّم بـ AUC و recall والـ confusion matrix مش بالـ accuracy لوحدها.

س: بتمنعوا الـ overfitting إزاي؟

ج: augmentation، transfer learning وقت التدريب بس، early stopping، LR scheduler، weight decay على validation AUC، واسترجاع أحسن checkpoint مش آخر epoch.

س: الـ metaheuristic optimization نفع؟

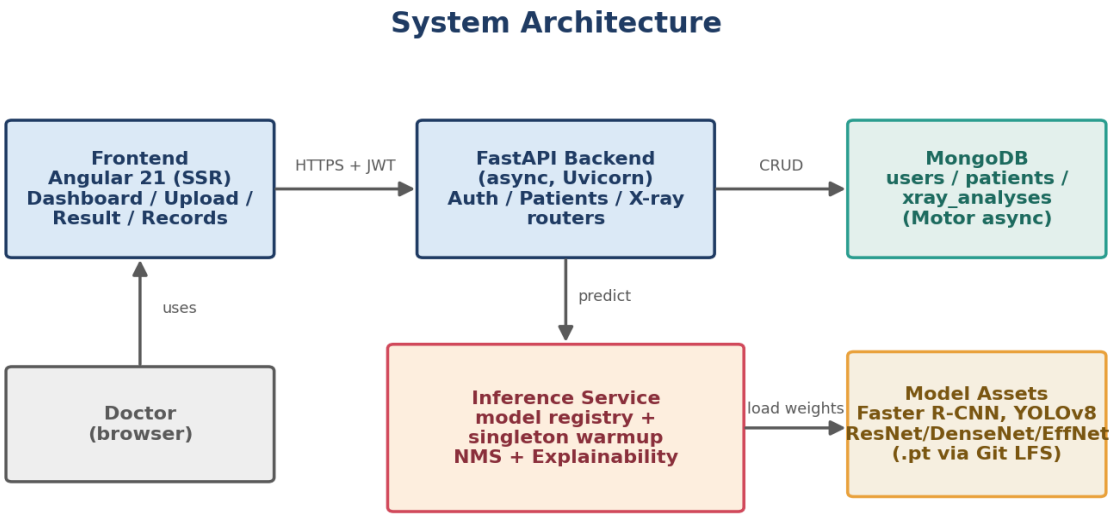
ج: مع الـ proxy الرخيص لأ: PSO و GWO و SA اختاروا إعدادات مكسبتش بعد retrain كامل. عشان كده اخترنا أحسن validation checkpoint، وضفنا قسم re-run أقوى يعيد البحث ببدجت ودقّة أعلى، وبيستبدل الـ model بس لو كسب الـ baseline.

س: الـ model آمن إكلينيكيًا؟

ج: هو decision support مش تشخيص أوتوماتيك. الدكتور بيأكد كل نتيجة، وكل تنبؤ بييجي بنسبة ثقة وheatmap عشان يتراجع. محتاج external validation وموافقات تنظيمية قبل الاستخدام الإكلينيكي الفعلي.

القسم B - الـ Web Application Pipeline

التطبيق من ثلاث طبقات: frontend الدكتور بيشوفه، backend بيعمل الشغل، وdatabase بيخزّن كل حاجة. وجوّه الـ backend في inference service متخصص بيشغّل الـ AI models. القسم ده بيشرح كل جزء وبعدين بيتتبّع طلب واحد من الضغطة للنتيجة.

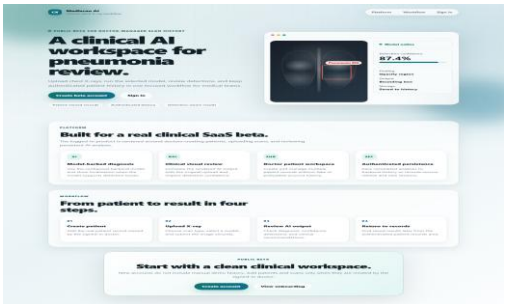


شكل B1. الـ three-tier architecture مع الـ inference service.

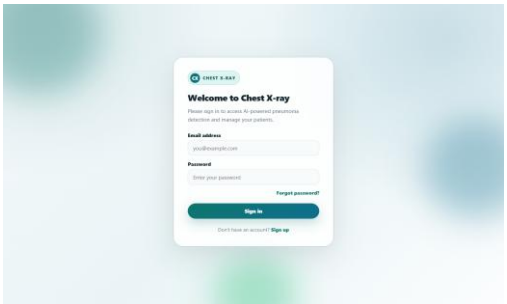
B1. الـ Frontend (Angular 21)

الـ frontend هو single-page application (SPA) متعمول بـ Angular ومكتوب بـ TypeScript. الـ single-page معناها إن المتصفح بيحمّل تطبيق واحد وبعدين بيبدّل الشاشات من غير reload كامل، فبيحس بالسرعة. مكوّناته:

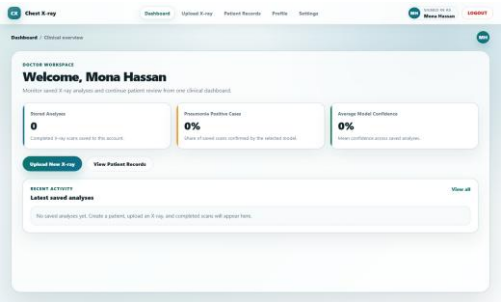
- **Components / pages:** login، signup، dashboard، upload، processing، result، patient records، profile، settings. كل واحدة قطعة UI مستقلة.
- **Routing + guards:** الـ router بيربط الـ URLs بالصفحات؛ والـ route guards بيمنعوا الصفحات المحمية (زي الـ dashboard) إلا لو المستخدم عامل login، وبيرجّعوا اللي عامل login بعيد عن صفحة الـ login.
- **State service:** analysis-state service مركزي بيمسك التحليل الحالي والـ history عشان الصفحات تشارك الـ data من غير ما نمررها بإيدينا.
- **API service:** service واحد بيجمّع كل النداءات للـ backend، وبيحط الـ login token مع كل request، فباقي التطبيق مبيتعاملش مع الـ HTTP مباشرة.
- **Server-side rendering (SSR):** الصفحات العامة بتترسم مبدئيًا على Express server صغير عشان أول ظهور يبقى سريع، وبعدين التطبيق بيعمل hydrate ويبقى SPA تفاعلي.



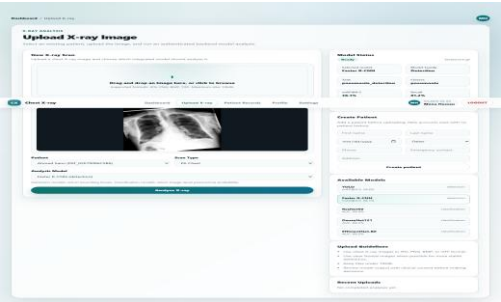
صفحة الترحيب (welcome)



صفحة الـ login



الـ dashboard



صفحة رفع الأشعة (upload) مع اختيار الـ model

B2. الـ Backend (FastAPI)

الـ backend عبارة عن FastAPI service بالـ Python. FastAPI غير متزامن (asynchronous)، يعني بيقدر يخدم طلبات كتير في نفس الوقت من غير ما يتعطّل: وهو مستني الـ database أو الـ model، بيخدم غيره. مكوّناته:

- **Routers:** الـ API متقسّم حسب المورد لـ auth و patients و x-ray، وده بينظّم الكود.
- **Pydantic models:** كل request و response بيتحقّق منه ضد schema بأنواع محددة، فالبيانات الغلط بترفض أوتوماتيك برسالة واضحة.
- **Lifespan startup:** لما الـ server يقوم بيتوصّل بـ MongoDB ويعمل warmup للـ default model (يحمّله في الذاكرة مرة واحدة)، وبينضّف عند الإغلاق.
- **Model warmup (singleton):** الـ model بيتحمّل مرة واحدة عند الإقلاع وبيتعاد استخدامه لكل طلب، فالمستخدم مبيدفعش تكلفة التحميل (ثواني) كل مرة. ده أهم قرار للأداء.

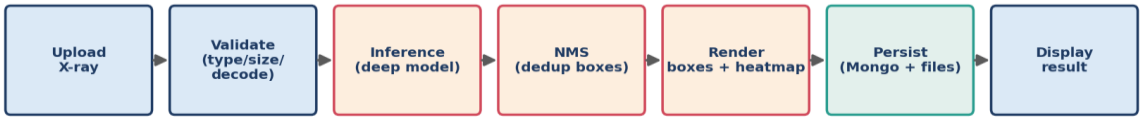
B3. الـ Inference Service خطوة بخطوة

ده قلب الـ backend. لما الصورة توصل، بيعمل:

- **Validate:** نوع الملف مسموح (.jpg/png/...)، الحجم أقل من 10MB، وإنه فعلًا بيتفكّ كصورة.
- **Model registry lookup:** بيدوّر على الـ model في registry (جدول بيربط مفتاح الـ model بالـ weights والـ task والـ thresholds). الـ registry فيه دلوقتي 5 models: YOLO، Faster R-CNN (detection)، ResNet50، EfficientNet-B0 (classification)، DenseNet121.
- **Predict:** بيشغّل الـ model عشان يطلّع التنبؤات الخام (boxes + scores، أو احتمال class).
- **Non-Maximum Suppression (NMS):** الـ detectors بيطلّعوا كذا box متداخل لنفس المنطقة؛ الـ NMS بيسيب أعلى box ثقة ويشيل اللي بيتداخل معاه أكتر من 30%، فالدكتور يشوف box واحد نضيف لكل منطقة. (Faster R-CNN بيعمل NMS داخليًا، فبنطبّقه على YOLO بس.)
- **Render + XAI:** بيرسم الصورة المعلَّمة (annotated) وheatmap الـ explainability.
- **Respond:** بيرجّع نتيجة منظّمة: القرار، الـ boxes، الثقة، وقت المعالجة، ومسارات الصور، وبعدين الـ backend بيحفظها.

الـ confidence thresholds: الـ detection فوق 0.10 بيتعرض كـ 'مشتبه' (suspected)؛ وفوق 0.25 بيتعامل كـ finding مؤكّد. الـ thresholds دي قابلة للضبط لكل model في الـ registry.

Prediction Data Flow



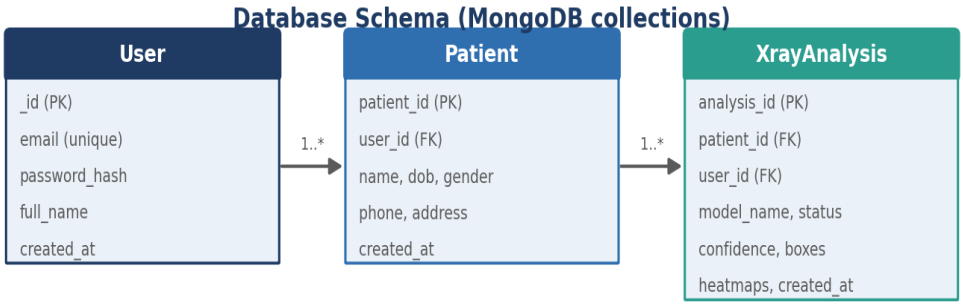
شكل B2. مسار البيانات للتنبؤ (prediction data flow).

B4. الـ Security

- **JWT authentication:** عند الـ login السيرفر بيصدر JSON Web Token موقّع (تذكرة مضادة للتزوير وبتنتهي). المتصفح بيبعته مع كل request، والسيرفر بيتأكد من التوقيع عشان يعرف إنت مين من غير sessions على السيرفر.
- **bcrypt password hashing:** الباسوردات مبتتخزنش plaintext أبدًا. bcrypt بيضيف salt عشوائي وبطيء بقصد، فده بيصعّب كسر قواعد بيانات الباسوردات المسروقة.
- **Per-user authorization:** كل database query متفلتر بـ id الدكتور المسجّل، فمفيش دكتور يقدر يقرأ مرضى دكتور تاني. ده متختبر.
- **Input validation + CORS:** الـ uploads بتتفحص قبل ما توصل لأي model، والـ API بيقبل طلبات من origins معروفة بس.
- **Production secret guard:** السيرفر بيرفض يقوم في الـ production بالـ secret key الافتراضي، فبيمنع غلطة نشر شائعة وخطيرة.

B5. الـ Database (MongoDB)

بنستخدم MongoDB، وهو document database، من خلال الـ Motor driver غير المتزامن. فيه ثلاث collections: users و patients و xray_analyses. اخترنا document database لأن التحليل الواحد طبيعي إنه يبقى document واحد مكتفي بذاته (الـ boxes والـ scores ومسارات الـ heatmaps والـ timestamps بنقراهم مع بعض دايمًا)، وده بيتطابق مع document شبه JSON ويتجنّب الـ joins المعقّدة. وفي unique indexes بتمنع تكرار الإيميلات والمعرّفات، وcompound index على المالك + وقت الإنشاء بيخلّي تحميل history الدكتور سريع.



شكل B3. الـ Entity Relationship Diagram للـ collections.

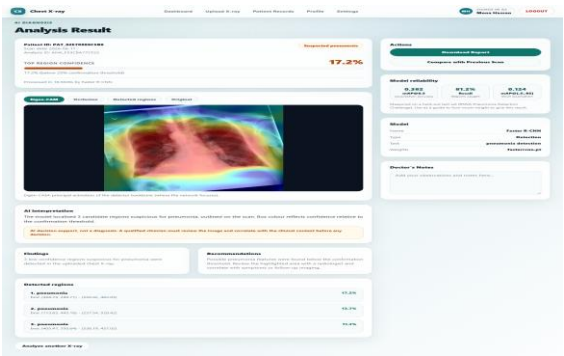
B6. الـ API

بيعمل إيه	Method	Endpoint
تسجيل، login (بيرجّع token)، بيانات المستخدم الحالي	POST/GET	api/auth/signup, /login, /me/
إنشاء/قراءة/تعديل/حذف المرضى (لكل دكتور)	CRUD	api/patients/
رفع صورة، تشغيل inference، حفظ النتيجة	POST	api/xray/upload/
عرض أو إدارة التحاليل المحفوظة	GET/DELETE	api/xray, /api/xray/{id}/

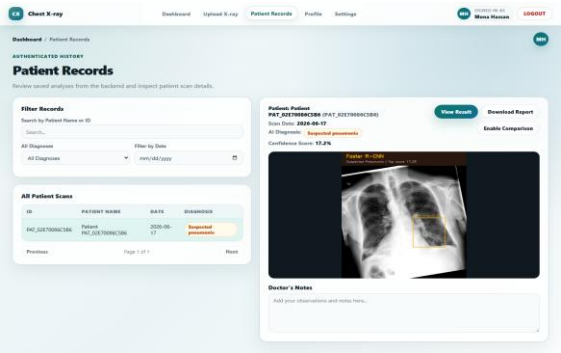
بيعمل إيه	Method	Endpoint
فحص الصحة؛ توثيق Swagger تفاعلي	GET	health, /docs/

B7. رحلة طلب كاملة من الأول للآخر

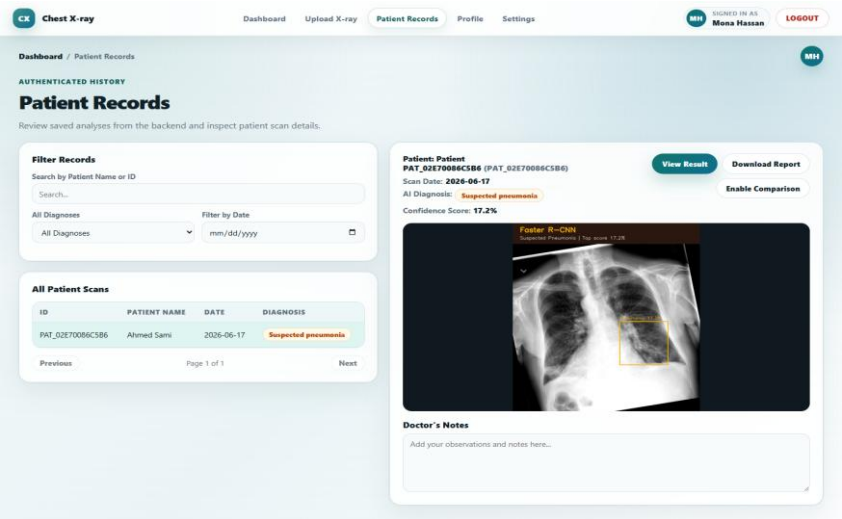
1) الدكتور بيعمل login؛ الـ frontend بيخزّن الـ JWT. 2) بيفتح مريض ويرفع أشعة؛ صفحة الرفع بتوريه preview ويختار الـ model. 3) الـ API service بيبعت الصورة (مع الـ token) لـ FastAPI/ api/xray/upload. 4) بيتأكد من الـ token، بيفحص الملف، ويسلّمه للـ inference service. 5) الـ model المحمّل بيتنبأ؛ الـ NMS بينضّف الـ boxes؛ وheatmap بيترسم. 6) النتيجة بتتكتب في collection الـ xray_analyses والصورة المعلَّمة بتتحفظ. 7) الـ backend بيرجّع النتيجة المنظّمة؛ والـ frontend بيعرضها في شاشة النتيجة ويضيفها لـ history المريض.



شاشة النتيجة: القرار + الـ box + نسبة الثقة + التوصيات



سجل التحاليل (records) لكل مريض



شكل B4. تفاصيل تحليل محفوظ (record detail) بكل النواتج.

B8. الـ Deployment والإعداد

- **الإعدادات** بتعيش في environment variables (الـ database URL، الـ secret key، الـ allowed origins، مدة الـ token)، وبعيدة عن الـ version control.
- **الـ Model weights** بتتخزّن بـ Git LFS لأنها ملفات binary كبيرة، فالـ repository يفضل صغير والـ clone بيجيبها عند الحاجة.
- **Scaling:** الـ API مبيمسكش حالة لكل مستخدم في الذاكرة (كل الحالة في الـ database والـ token)، فينفع ينتسخ ورا load balancer؛ والـ model ممكن يتنقل لـ service لوحده للأحمال الثقيلة.
- **التشغيل:** شغّل MongoDB، شغّل الـ backend بـ Uvicorn، ابني وقدّم الـ Angular frontend؛ الخطوات الكاملة في الـ README ودليل المستخدم في الـ thesis.

B9. القسم B - أسئلة متوقّعة من اللجنة

س: ليه FastAPI و async؟

ج: الـ inference والـ database calls بتتضمّن انتظار؛ السيرفر الـ async بيفضل يخدم باقي المستخدمين أثناء الانتظار، فالتطبيق يفضل سريع تحت الضغط من غير thread لكل request.

س: ليه MongoDB مش SQL؟

ج: التحليل الواحد document متداخل مكتفي بذاته بنقراه كوحدة، وده بيناسب الـ document store ويتجنّب الـ joins. وبرضه بنفرض البنية بـ indexes و validation.

س: الـ login بيشتغل إزاي بالظبط؟

ج: عند الـ login بنتأكد من الباسورد المعمول له bcrypt-hash وبنصدر JWT موقّع بمدة صلاحية. المتصفح بيبعت الـ token مع كل request، والسيرفر بيتأكد من التوقيع عشان يعرف المستخدم من غير sessions.

س: بتمنعوا دكتور يشوف مرضى دكتور تاني إزاي؟

ج: كل query متفلتر بـ id الدكتور المسجّل، فالسجلات اللي مش بتاعته بترجع not-found. وعندنا automated test بيثبت إن الوصول عبر المستخدمين مرفوض.

س: ليه بتحمّلوا الـ model مرة واحدة عند الإقلاع؟

ج: تحميل الـ model بياخد ثواني؛ لو عملناه لكل request هيخلّي التطبيق بطيء. بنحمّله مرة (warmup) ونعيد استخدامه، فكل تنبؤ سريع والذاكرة متوقّعة.

س: الـ NMS ده إيه وليه محتاجينه؟

ج: الـ Non-Maximum Suppression بيشيل الـ boxes المكررة المتداخلة، ويسيب أعلى واحد ثقة لكل منطقة (هنا لما التداخل يعدّي 30%)، فالدكتور يشوف box واحد نضيف بدل كذا واحد.

س: بتأمّنوا الـ uploads إزاي؟

ج: بنتأكد من النوع والحجم وإن الملف بيتفكّ كصورة قبل ما يوصل model؛ والـ API ورا authentication و CORS؛ والـ secrets في environment variables مش في الكود.

س: هتعملوا scale للمستشفى إزاي؟

ج: الـ API stateless، فبنشغّل كذا نسخة ورا load balancer، وننقل الـ model لـ inference service أو GPU server مخصص، ونستخدم managed MongoDB cluster. الـ containerization والـ monitoring هما أول خطوة جاية.