**تقرير لجنة الكود**

**تقديم: محمد نور علي**

المشرفون: د. رياض سنبل، م. محمد بشار دسوقي

بناء منصة تواصل للاستشارات مدعومة بالذكاء الصنعي

# الفصل الأول

التعريف بالمشروع

يتضمن هذا الفصل التعريف بالمشروع ومتطلباته.

## مقدمة

خلال العقود القليلة الماضية، تسارعت وتيرة تطوّر تقنيّات الذكاء الصنعي (AI) بصورة غير مسبوقة؛ فبعد حقبة نماذج التعلّم التقليدي (ML) ظهر جيل جديد من نماذج اللغات الكبيرة (LLMs) القادرة على فهم السياق، تحليل النصوص متعددة اللغات، وتصميم استجابات شبيهة بالبشر بدقّة لافتة. انعكس هذا التطوّر على مجالات واسعة؛ فقد أصبحنا نرى حلولًا ذكيّة في التشخيص الطبّي المبكر، وتحليل العقود القانونيّة، وإدارة خدمة العملاء عبر المساعدات الافتراضيّة (Virtual Assistants) والدردشات الآليّة (Chatbots). كما دخل الذكاء الصنعي بقوّة في دعم قرارات الأعمال.

على الرغم أنّ منصّات الذكاء الصنعي أثبتت كفاءتها في إنتاج حلول سريعة وقابلة للتوسّع، فإنّ الاعتماد الكامل على الخوارزميّات يظلّ محفوفًا بالمخاطر عندما يتعلّق الأمر بسلامة المرضى أو دقّة الاستشارات القانونيّة؛ إذ تتطلّب هذه المجالات دقّة عالية ومساءلة أخلاقيّة لا توفّرها الخوارزميّات بمفردها. يُطرح هنا سؤال محوري: كيف نوفّق بين سرعة الأتمتة التي يوفّرها الذكاء الصنعي والموثوقيّة التي يجلبها الإنسان الخبير؟

ومن هنا برزت الحاجة إلى منصّات هجينة تجمع بين قدرات الذكاء الصنعي في معالجة البيانات الضخمة واقتراح الحلول الأوليّة، وبين دور المستشارين المتخصّصين في التدقيق، والتكييف، وتحمل مسؤولية القرار بخبرته المهنيّة.

## الهدف من المشروع

يندرج يقدّم مشروعنا «منصّة الاستشارات الذكيّة» حلًّا وسطيًّا يجمع مزايا الطرفين. حيث يمكن تقسيم العمل إلى مرحلتين المرحلة الأولى للاستشارة—كاستقبال الطلب وفهمه مبدئيًّا والبحث في المراجع—تُعهد إلى الذكاء الصنعي لضمان الاستجابة الفوريّة وتقليص زمن الانتظار. أمّا المرحلة الحاسمة، أي اختيار الحلّ النهائي وصياغته، فتبقى في يد المستشار المختصّ الذي يضفي على المخرجات البُعد الإنساني ومعايير السلامة المهنيّة. بذلك نحصل على مزيج يجمع:

السرعة والإتاحة: المعالجة الرقميّة تعني أنّ المستخدم ليس مضطرًّا لحجز موعد أو زيارة مقرّ فعليّ، بينما يسهّل الـAI البحث في مراجع ضخمة خلال ثوانٍ.

الموثوقيّة والمحاسبة: مرور الحلول عبر خبير مؤهَّل يضمن الترشيح الأخلاقي والتأكّد من توافق التوصيات مع الممارسات المهنيّة السليمة.

التخصّص القابل للتوسّع: بنية الخدمات المصغّرة (Microservices) تتيح إضافة مجالات استشاريّة جديدة أو دمج مصادر معرفة محدَّثة دون التأثير في استقرار المنظومة.

التتبّع والشفافيّة: يُسجَّل كل تفاعل في السجلّات، مع إمكان تدقيق القرارات لاحقًا واستثمار البيانات في تحسين النماذج.

بهذا التصميم الهجين، نلبّي حاجة ملحّة للسوق: منصّة تستطيع استغلال قدرات الذكاء الصنعي المتقدّمة دون التفريط في ثقة المستخدمين ومعايير السلامة، مع الحفاظ على تجربة استخدام سلسة وشخصيّة.

ما ذكر سابقاً يمكن أن ندرجه في جزء الهدف من المشروع ولكن لنجعله مختصر بشكل أكبر.

## المتطلبات الوظيفية

يجب أن يقدم النظام للمستشير (الشخص الذي يطلب الاستشارة) ما يلي:

1. السماح بطلب استشارة تتعلق بحالة (قضية) معينة باختصاصات مختلفة.

يجب أن يقدم النظام للمستشار (الشخص الذي يقوم بمعالجة طلب الاستشارة) ما يلي:

1. السماح بإنشاء حساب جديد ضمن النظام.
2. السماح بتسجيل الدخول من حساب مُنشأ سابقاً.
3. السماح باستعراض الحالات المسندة له.
4. السماح باستعراض اقتراحات (توصيات) الذكاء الصنعي كحلّ لهذه الحالة.
5. السماح بمعالجة حالة مسندة له.
   1. السماح باختيار واحد من اقتراحات الذكاء الصنعي واعتماده كحلّ لهذه الحالة.
   2. السماح بإضافة حل جديد لهذه الحالة واعتماده.
6. السماح بإضافة مراجع جديدة إلى مصادر المعرفة الخاصة بالذكاء الصنعي.

يجب أن يقدم النظام لمدير النظام ما يلي:

1. السماح بقبول أو رفض طلب إنشاء حساب لمستشار.
2. السماح بإضافة تخصصات جديدة للحالات التي يمكن للمنصة معالجتها.

## المتطلبات غير الوظيفية

1. يجب أن يكون النظام آمناً، حيث يسمح فقط للمستخدمين المسجّلين باستخدامه.
2. يجب أن يوفر النظام واجهات مرنة، سهلة الاستخدام وجيدة المظهر.
3. يجب أن يكون الكود البرمجي قابلاً للتعديل والصيانة.
4. يجب أن يكون النظام قابلاً للتوسع بسهولة.
5. يجب أن يكون النظام قابلاً للنشر بشكل ناجح على بيئتين سحابيتين مختلفتين دون تعديل الكود.
6. يجب أن تكون نتائج اقتراحات الذكاء الصنعي دقيقة بحد كافي يضمن اعتمادها من قبل المستشارين دون تعديلات جوهريّة في أغلب الحالات: أكثر من 85% من المقترحات التي ينتجها الذكاء الصنعي تعتمد كما هي أو بعد تعديلات طفيفة وذلك عند أخذ عيّنة لا تقل عن 200 حالة.

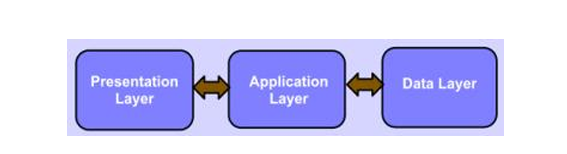
# الفصل الثاني

الدراسة النظرية

يوضّح هذا الفصل مجموعة من المفاهيم النظرية المستخدمة ضمن العمل المقدم.

## البنية المعمارية 3-tier:

النموذج ثلاثي الطبقات يحتوي عادةً على المكونات البرمجية التالية المذكورة من الطبقة العليا إلى الطبقة السفلية: طبقة العرض، طبقة التطبيق وطبقة البيانات، كما هو موضح في الرسم البياني []. يمكن للطبقة الوصول بشكل مباشر فقط إلى المكونات العامة للطبقة التي تقع مباشرةً أدناها. على سبيل المثال، يمكن لطبقة العرض الوصول فقط إلى المكونات العامة في طبقة التطبيق، ولكن ليس في طبقة البيانات. وكما هو الحال بالنسبة لطبقة التطبيق التي يمكنها الوصول فقط إلى المكونات العامة في طبقة البيانات، ولكن ليس في طبقة العرض. إن القيام بذلك يمكن أن يقلل من الاعتمادات بين طبقة وأخرى. إنّ تقليل هذه الاعتمادات سيجلب فوائد لتطوير/صيانة الطبقات، التوسع، وما إلى ذلك. كما أن القيام بذلك يجعل فرض أمان الطبقات ممكناً. على سبيل المثال، لا يمكن للطبقة التي يتعامل معها العميل (أي طبقة العرض) الوصول إلى طبقة البيانات مباشرةً ولكن من خلال طبقة التطبيق، لذا فإن طبقة البيانات محمية بشكل أفضل. أخيراً، يمكن أن يساعد القيام بذلك أيضاً في تجنب الاعتماديات الدائرية بين مكونات البرمجيات



الطبقات الثلاث الموضحة في المخطط موصوفة بإيجاز كما يلي:

* طبقة العرض: هي طبقة يمكن للمستخدمين الوصول إليها مباشرة مثل واجهة سطح المكتب، صفحة Web، وما إلى ذلك. تُعرف أيضاً باسم طبقة العميل.
* طبقة التطبيق: تحتوي هذه الطبقة على منطق الأعمال مثل قواعد الأعمال والتحقق من البيانات، ومفاهيم المجال، ومنطق الوصول إلى البيانات وما إلى ذلك. تُعرف أيضاً باسم الطبقة الوسطى.
* طبقة البيانات: مصدر البيانات الخارجي لتخزين بيانات التطبيق، مثل خادم قاعدة البيانات، نظام إدارة علاقات العملاء، نظام تخطيط موارد المؤسسات، الإطار الرئيسي أو أنظمة قديمة أخرى وما إلى ذلك.

## ممارسة التكامل المستمر والنشر المستمر في تطوير البرمجيات (CI/ CD)

يعد التكامل المستمر (CI) والنشر المستمر (CD) من الممارسات الأساسية في تطوير البرمجيات الحديثة، وخاصة ضمن أطر عمل DevOps. يعمل CI/CD على أتمتة عمليات اختبار التطبيقات ونشرها، مما يمكن الفرق من تقديم التحديثات بسرعة وكفاءة بعد أي تغيير يحدث ضمن الكود البرمجي (source code).

## الخدمات المصغرة (Microservices)

تمثل الخدمات المصغَّرة أسلوباً معمارياً حديثاً يحلل التطبيقات التقليدية ذات الكتلة الواحدة (monolithic applications) إلى مجموعة من الخدمات الصغيرة الموزعة القابلة للنشر بشكل مستقل عن بعضها البعض. يعزز هذا النهج المرونة في بناء التطبيقات وقابلية التوسع ويزيد من إنتاجيَّة المطورين من خلال السماح للفِرق بالعمل بشكل مستقل على خدمات مختلفة بدعم من ممارسات DevOps وCI / CD.

تتميَّز الخدمات المصغرة بتماسكها الداخلي العالي (high cohesion) وارتباطها الفضفاض (loose coupling)، مما يسهل تطوير التطبيقات المعقَّدة من خلال واجهات برمجة تطبيقات (APIs) بسيطة ومحددة جيداً.

## نماذج اللغات الكبيرة (LLM)

هي نماذج لغوية إحصائية تستفيد من تقنيات التعلم العميق وخاصةً هياكل المحولات لفهم اللغة البشريّة وتوليدها. تتميز هذه النماذج بحجمها الكبير، وغالباً ما تحتوي على عشرات إلى مئات المليارات من المعاملات (parameters)، إذ يتم تدريبها على كميّات هائلة من بيانات النصوص من مصادر متنوعة مثل الكتب ومواقع الويب وبيانات المحادثة.

كمثال على هذه النماذج سلسلة GPT المقدمة من شركة OpenAI، PALM المقدمة من Google و LLAMA من شركة Meta.

## هندسة الأوامر Prompt Engineering

الأمر أو المحفّز (Prompt) هو نص يتم تقديمه لنموذج لغوي لمساعدته على توليد استجابة.

هندسة الأوامر (Prompt engineering) هي عملية صياغة وتحسين الأوامر للتواصل بشكل فعال مع نماذج اللغات الكبيرة (LLMs). هذه العملية مهمّة للحصول على ردود دقيقة وذات صلة من النموذج. مع تطور نماذج اللغة، أصبحت مهارة هندسة الأوامر أساسية للمستخدمين الذين يريدون الاستفادة القصوى من نماذج اللغات الكبيرة وتحقيق أفضل النتائج في مختلف المجالات.

عند تصميم هذه الأوامر يجب مراعاة المعايير التالية:

* الوضوح: يجب أن تكون الأوامر واضحة وسهلة الفهم، حيث يساعد ذلك على توليد استجابة أكثر دقة من قبل النماذج اللغوية (LLMs) [5].
* إضافة قيود صريحة: يجب إضافة إرشادات وقيود محددة عند الطلب، حيث يساعد ذلك في تضييق نطاق تركيز النموذج اللغوي مما يؤدي إلى استجابة ذات صلة بالطلب [5].
* التجريب: يجب تجريب أنواع مختلفة من الأوامر لمعرفة ما هو الأفضل، حيث أن تجربة تنسيقات مختلفة يمكن أن يساعد في اكتشاف طرق فعالة للتفاعل مع النموذج اللغوي [5].
* تحسين الأوامر باستمرار: يجب الاستمرار في تحسين الأوامر بناءً على النتائج التي يعيدها النموذج اللغوي، حيث أن هذه العملية التكرارية يمكن أن تعمل على تحسين الأوامر بشكل كبير بمرور الوقت [5].
* التحكم بمعاملات النموذج: يمكن أن يؤدي تغيير معامل مثل درجة الحرارة الخاصة بالنموذج (temperature) التي تحدد مدى إبداع النموذج إلى نتائج مختلفة، كما أن استخدام سلسلة من الأوامر واحدة تلو الأخرى يؤدي إلى إنشاء تفاعلات أكثر تعقيداً مع النموذج [5].
* تقديم معلومات إضافية للأمر: إن إضافة سياق المهمة المطلوبة إلى الأوامر يمكن أن تساعد في إنتاج استجابات أكثر دقة وملاءمة. يكون هذا مفيدًا بشكل خاص عند التعامل مع مفاهيم مجردة أو مجالات متخصصة [5].
* تقديم أمثلة: يمكن أن يساعد إضافة مجموعة من الأمثلة للخرج المتوقع (Few-shot learning) ضمن الأمر على الوصول لخرج أكثر دقة، حيث أن التعلم من خلال عدد قليل من الأمثلة يجعل النماذج قابلة للتكيف، خاصةً في السيناريوهات ذات البيانات المحدودة، كما أن هذا الأسلوب يقلل من الإفراط في التجهيز(overfitting) ويعزز المرونة والتخصيص والتكيف السريع مع المهام الجديدة [6].

## التوليد المعزز بالاسترجاع Retrieval – Augmented Generation (RAG)

يستخدم نظام RAG قاعدة المعرفة الخارجية 𝑫 لتعزيز نظام التوليد. مثلاً بأخذ المستندات الخارجية، تتكون 𝑫 من مستندات خارجية، يحتوي كل منها على مجموعة من المقاطع chunks 𝑐𝑖∈𝑪𝒊. يتم تحويل هذه المقاطع إلى تمثيل شعاعي باستخدام نموذج تضمين معين. عند إدخال استعلام 𝑞، الذي يتم أيضاً تضمينه كمتجه، يقوم المسترجع في نظام RAG باسترجاع أعلى 𝑘 مقاطع 𝑅𝑞={𝑟1,𝑟2,...,𝑟𝑘} الأكثر صلة بالاستعلام 𝑞. يمكن لنظام RAG استخدام طرق دمج مختلفة لدمج المقاطع المسترجعة.

تتم صياغة العملية العامة على النحو التالي:

كما هو موضح في الشكل []، تتكون RAG عادة من ثلاث مكونات، الاسترجاع، والتوليد، ودمج الاسترجاع.

عادةً ما يتكون وحدة الاسترجاع من ثلاث مكونات: مشفر encoder لتشفير المدخلات إلى تضمينات embeddings، وفهرسة فعالة تدعم البحث عن الجيران الأقرب تقريبياً، ومخزن بيانات لتخزين المعرفة الخارجية في شكل أزواج مفتاح-قيمة. التحدي الرئيسي في وحدة الاسترجاع هو إيجاد التوازن الأمثل بين كفاءة الاسترجاع وجودة الاسترجاع. تشير كفاءة الاسترجاع إلى مدى سرعة الحصول على المعلومات ذات الصلة، مما ينطوي على تسريع التشفير، والفهرسة الفعالة، إلخ. تشير جودة الاسترجاع إلى مدى ملاءمة المعلومات التي يمكن استرجاعها، مما ينطوي على تعلم تمثيل المقاطع chunks، وخوارزميات البحث المتقدم عن الجيران الأقرب، إلخ [7].

تسعى تقنيات دمج الاسترجاع إلى الاستفادة من المعلومات المسترجعة لتعزيز عملية التوليد. يمكن تصنيف هذه التقنيات إلى ثلاثة أنواع رئيسية: دمج قائم على الاستعلام، ودمج كامن، ودمج قائم على النتائج (اللوغاريتمات). يضيف دمج الاستعلام مدخلات مع الاسترجاعات قبل إدخالها إلى مولدات النصوص، بينما يركز دمج النتائج على النتائج الناتجة عن مولدات النصوص ويدمج النتائج المسترجعة للحصول على نتائج أكثر قوة. أما الدمج الكامن فهو يعتمد على إدخال تمثيلات الاسترجاع إلى التمثيلات الكامنة لمولدات النصوص، وبالتالي تحسين أداء النماذج بشكل غير مباشر [7].

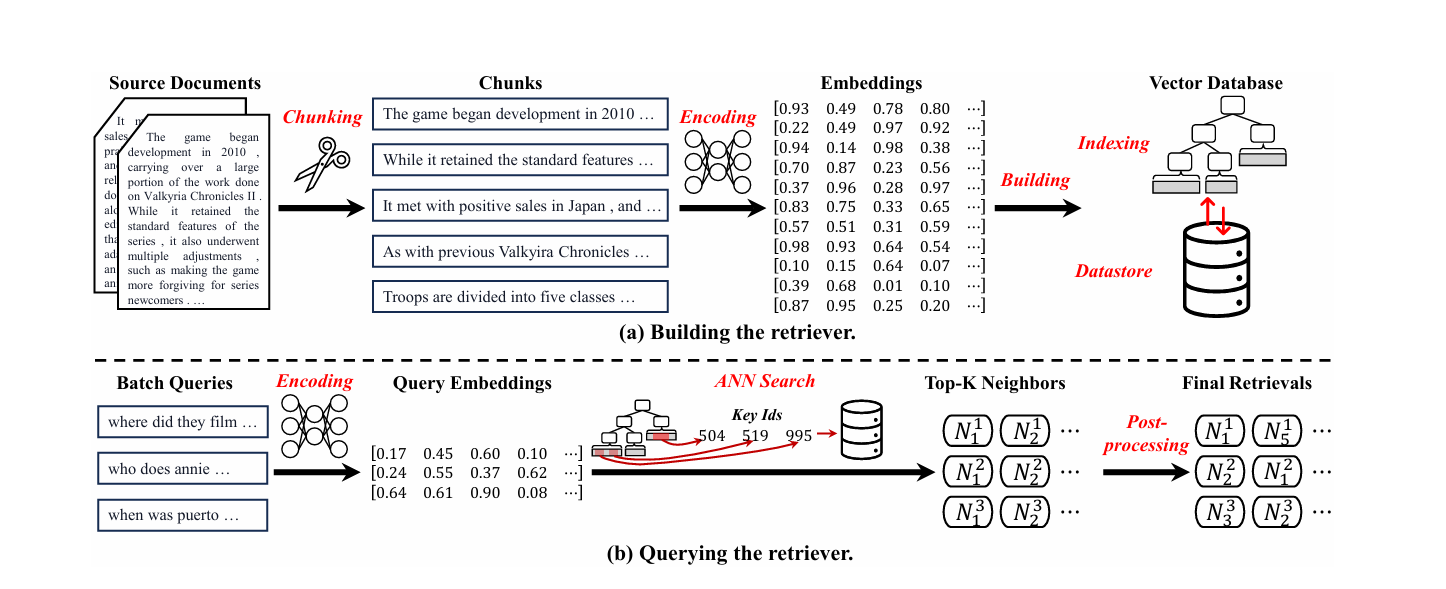
يمكن تصنيف وحدات المولدات Generators modules إلى فرعين من المولدات: المولدات الافتراضية Default Generator ومولدات التعزيز الاسترجاعي Retrieval Augmented (RA). تشمل المولدات الافتراضية معظم نماذج اللغة الكبيرة التي تم تدريبها مسبقاً / تحسينها، مثل نماذج سلسلة GPT، ونماذج Mistral، ونماذج سلسلة Gemini. تشير مولدات RA إلى المولدات التي تم تدريبها مسبقاً / تحسينها والتي تتكون من وحدات لدمج الاسترجاعات، مثل RETRO وEnc-Dec. تولد تلك المولدات ردود أو تقوم بعمل تنبؤات [7].

بشكل ملخص تشمل طريقة العمل في RAG ثلاث خطوات:

1) استرجاع المعلومات ذات الصلة من قواعد البيانات الخارجية بناءً على المدخلات المعطاة.

2) دمج المعلومات المسترجعة مع المدخلات أو الحالات الوسيطة بناءً على تقنيات الدمج.

3) تقديم توقعات من قبل المولدات استناداً إلى المدخلات والاسترجاعات المقابلة.



## JSON Web Token (JWT)

هو وسيلة خفيفة لتبادل البيانات بين طرفين لتسهيل المصادقة والتفويض والأمان. يتم تخزين كل عبارة JWT ككائن JSON ويتم استخدام كل كائن JSON كنص عادي في تشفير الويب JSON أو كحمولة لتوقيع الويب JSON (JWS) مما يسمح للادعاءات بأن تكون مؤمنة رقمياً ومصادق عليها مع رمز التوثيق برسالة (MAC). قبل بضع سنوات، كانت 'الرموز' مجرد سلاسل نصية بدون قيمة جوهرية قبل JWT، على سبيل المثال: 2pWS6RQmdZpE0TQ93X. تم التحقق من هذه الرموز لاحقاً في خادم حيث تم تخزين بيانات الرموز. العيب في هذا النهج هو أنه في كل مرة يتم فيها استخدام الرمز، تكون هناك حاجة للوصول إلى قاعدة البيانات (أو ذاكرة التخزين المؤقت). في الوقت الحاضر، تقوم JWTs بتشفير البيانات والتحقق من عباراتها الخاصة (عن طريق التوقيع). وبالتالي تم تطوير JWTs قصيرة الأمد. وبالتالي، لا حاجة لقاعدة بيانات. وبالتالي يتم القضاء على عبء قاعدة البيانات ويتم تبسيط التصميم لأنه يتعين على الخادم الذي يصدر رموز JWT التفكير فقط في إدخال قاعدة البيانات / طبقة الاستمرارية (رمز التحديث) [8].

# الفصل الثالث

الدراسة المرجعية

يعرض هذا الفصل الأبحاث والبيانات المرتبطة بالعمل المقدم.

## مقدمة

في السنوات الأخيرة ظهرت عدة منصّات وخدمات تستفيد من تقنيات الذكاء الاصطناعي لتقديم الاستشارات في مجالات متنوعة. يركّز كثير من هذه المنصّات على مجال اختصاصي محدّد، حيث تمزج بين قدرة نماذج اللغة الكبيرة (LLMs) على التوليد، وبين قواعد معرفية متخصصة لتقديم استشارات دقيقة للمستخدم. في هذه الدراسة المرجعية نستعرض بعض المنصّات البارزة المشابهة من المجال الطبي والقانوني، إضافة إلى لمحة عن الأبحاث الأكاديمية والتقنيات المتعلقة مباشرةً بمنصّة الاستشارات المقترحة.

## منصّات الاستشارات الطبية المدعوة بالذكاء الصنعي

منصة "تبيان" للمساعد الطبي الذكي – جامعة حلب:

تعتبر "تبيان" مثالًا متقدمًا على منصّات الاستشارات الطبية المدعومة بالذكاء الاصطناعي في العالم العربي. أُطلقت "تبيان" حديثًا في مستشفى حلب الجامعي كأول نظام سريري ذكي يعمل ميدانيًا في سوريا. تعتمد المنصّة على تقنيات الذكاء الاصطناعي الحديثة وبالتحديد منهجية التوليد المعزز بالاسترجاع (LLM + RAG) لمعالجة نتائج التحاليل المخبرية وتقديم تقارير طبية شاملة. يقوم النظام بتحليل نتائج الفحوصات المخبرية للمريض باستخدام نموذج لغة كبير مدعّم بمصادر معرفة طبية موثوقة مثل قواعد بيانات Mayo Clinic ومنظمة الصحة العالمية وUpToDate وغيرها.

بفضل هذه المراجع، يقوم تبيان بتوليد تقرير طبي بصيغة PDF يتضمّن شرحًا للنتائج غير الطبيعية، وملخّصًا سريريًا مبسّطًا، وتوصيات أوّلية مدروسة، بالإضافة إلى خطوات متابعة للحالة. يتميز التقرير بلغة مبسطة وأسلوب احترافي يساعد الأطباء في تسريع عملية التشخيص وتحسين تجربة المريض. الجدير بالذكر أنّ النظام تم تدريبه على أكثر من 3300 تحليل مخبري فعلي لتعزيز دقته، وقد أثبتت الاختبارات كون توصياته ذات موثوقية عالية وتم اعتماده رسميًا للاستخدام السريري. يمثل تبيان نموذجًا رائدًا للابتكار المحلي في تطبيق تقنيات الذكاء الاصطناعي الطبية، مع الحفاظ على وجود الطبيب في الحلقة للإشراف على المخرجات قبل اعتمادها.

منصة Babylon Health للاستشارة الصحية:

على الصعيد العالمي، برزت منصّة Babylon Health في المملكة المتحدة كمزود للخدمات الصحية الافتراضية بالاعتماد على الذكاء الاصطناعي. قدمت Babylon تطبيقًا تفاعليًا يعمل كـ مساعد افتراضي للأعراض الصحية، بحيث يستطيع المستخدم إدخال الأعراض التي يعانيها ويجيب عن سلسلة من الأسئلة التي يطرحها chatbot ذكي. يستخدم نظام Babylon تقنيات تعلّم عميق متقدمة طُوّرت بمساهمة أطباء ومختصين، مما يجعله أكثر من مجرد قاعدة بيانات بحثية. يقوم بتحليل الأعراض وعوامل الخطورة المعروفة لدى المريض ليقدم معلومات طبية مستندة إلى أحدث ما هو متاح من المعرفة. هدف هذه المنصة كان إجراء عملية فرز وتشخيص مبدئي (triage) لمساعدة المرضى على فهم حالاتهم الصحية وتقديم توصيات مبدئية، وبالتالي تخفيف العبء عن الأطباء وتقليل زمن الانتظار. بالفعل، تمكن التطبيق من تقديم آلاف الاستشارات يوميًا على مستوى عالمي. على الرغم من تحديات تتعلق بدقة بعض التشخيصات وانتقادات حول الاعتماد الكلي على الآلة، مثّلت Babylon إحدى أولى المحاولات الواسعة لتوظيف الذكاء الاصطناعي في الاستشارات الطبية عن بعد والاستفادة من تعلم الآلة في تقييم الحالات الصحية للمستخدمين.

## منصات الاستشارات القانونية المدعومة بالذكاء الصنعي

منصة "عدالة" للاستشارات القانونية الذكية – السعودية:

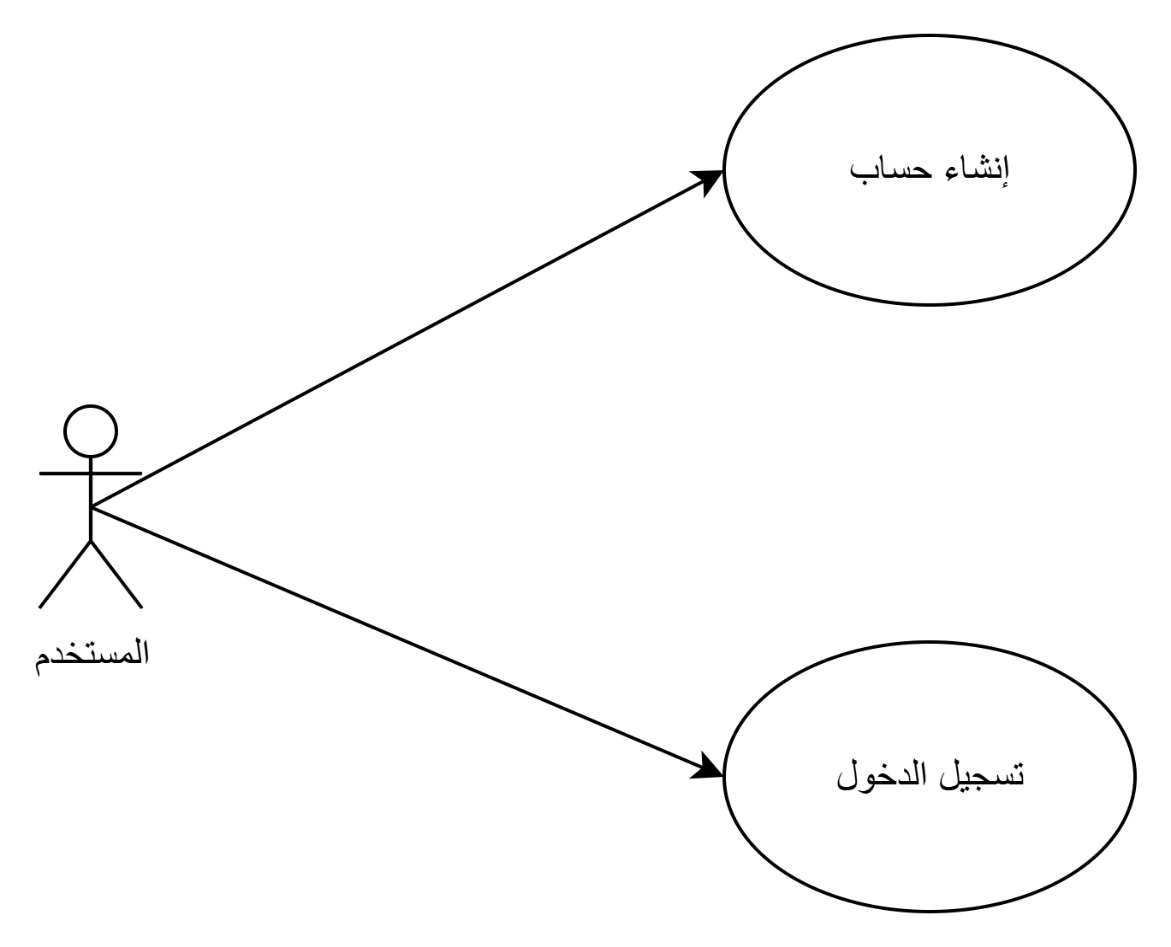
في المجال القانوني، ظهرت منصة "عدالة AI" في المملكة العربية السعودية لتقديم استشارات قانونية فورية بالاعتماد على الذكاء الاصطناعي. تتيح عدالة للمستخدم (سواء كان فردًا أو شركة أو محاميًا) الحصول على مشورة قانونية دقيقة في ثوانٍ دون الحاجة إلى انتظار أو إجراءات معقدة. تعتمد المنصة على نظام ذكاء اصطناعي متخصص في فهم الأنظمة السعودية، حيث يمكنه تحليل المستندات القانونية والعقود واستيعاب السياق النظامي لها. كما توفر المنصة مكتبة قانونية مُحدَّثة تربط المستخدم مباشرةً بأهم اللوائح والأنظمة والأحكام القضائية السعودية ذات الصلة بالاستشارة المطلوبة. يمتاز النظام بسهولة الاستخدام من خلال واجهة حوارية بسيطة، مما يسمح حتى لغير المختصين قانونيًا بالحصول على إجابات موثوقة لأسئلتهم.

# الفصل الرابع

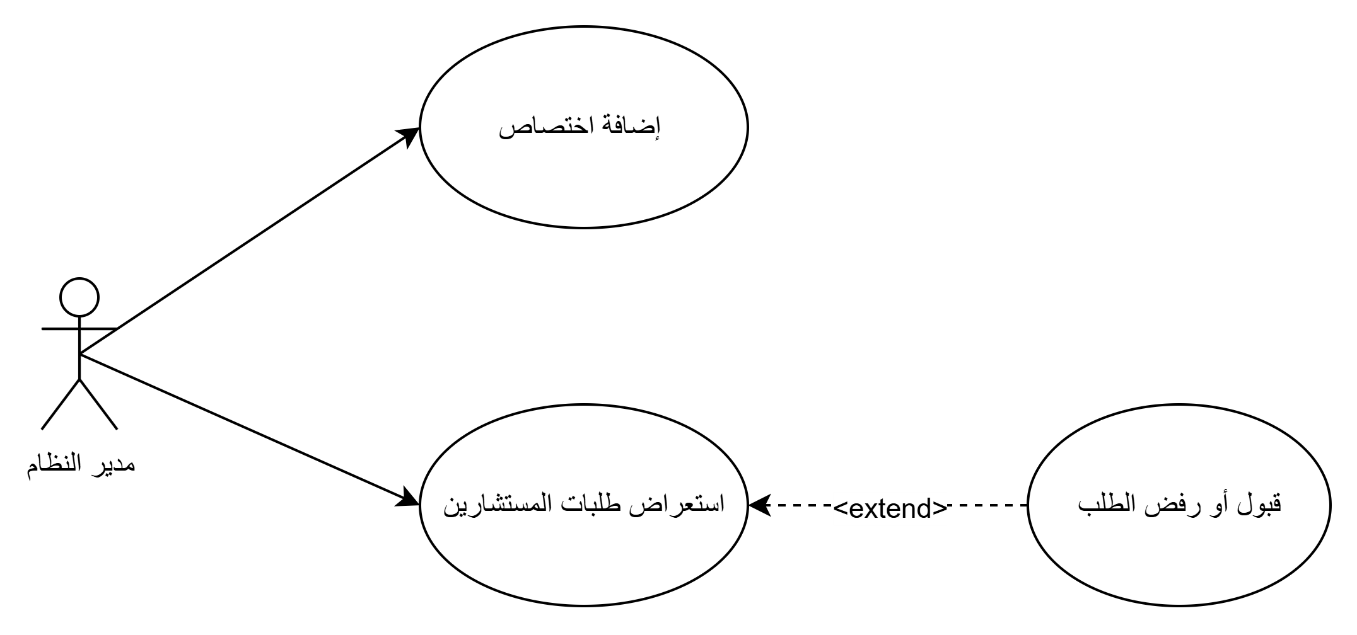
الدراسة التحليلية

يعرض هذا الفصل عمليّة تحليل النظام ودراسة متطلباته.

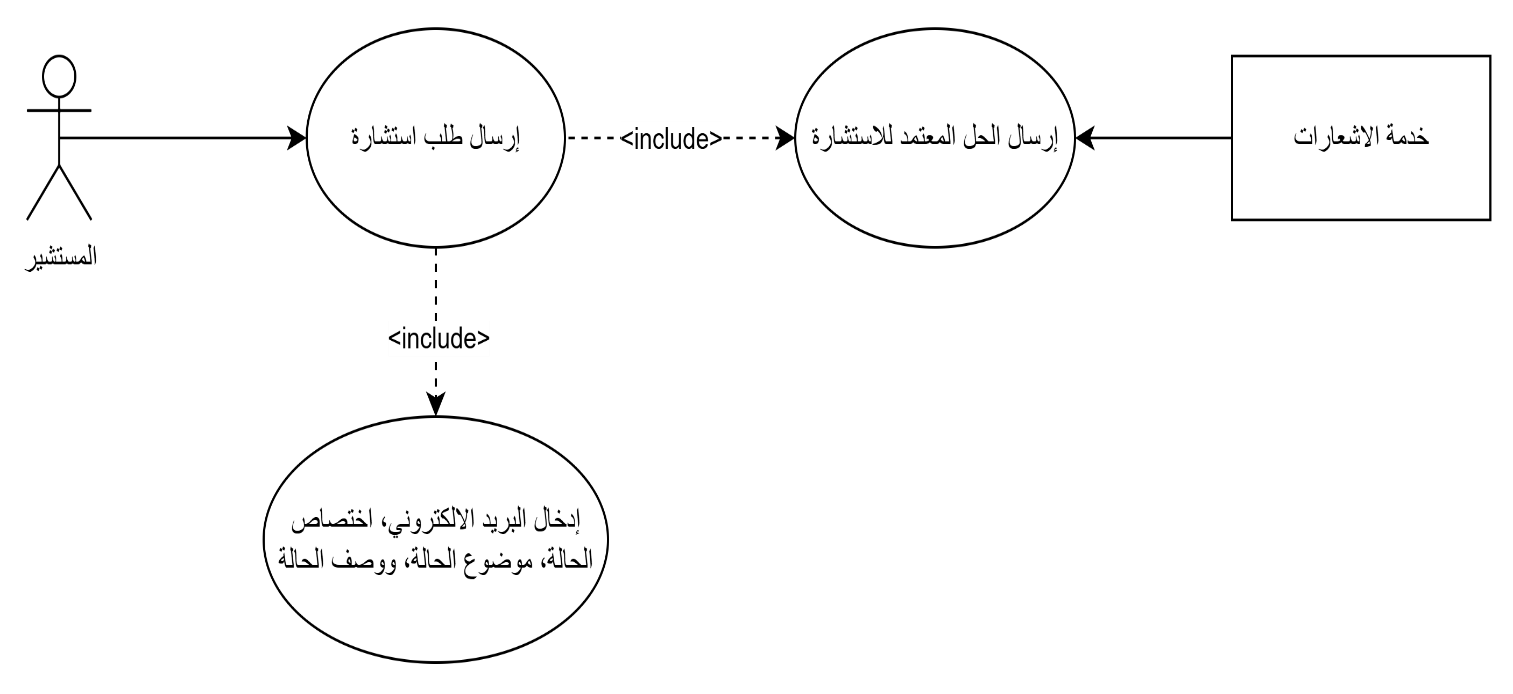
## مخططان حالات الاستخدام

**

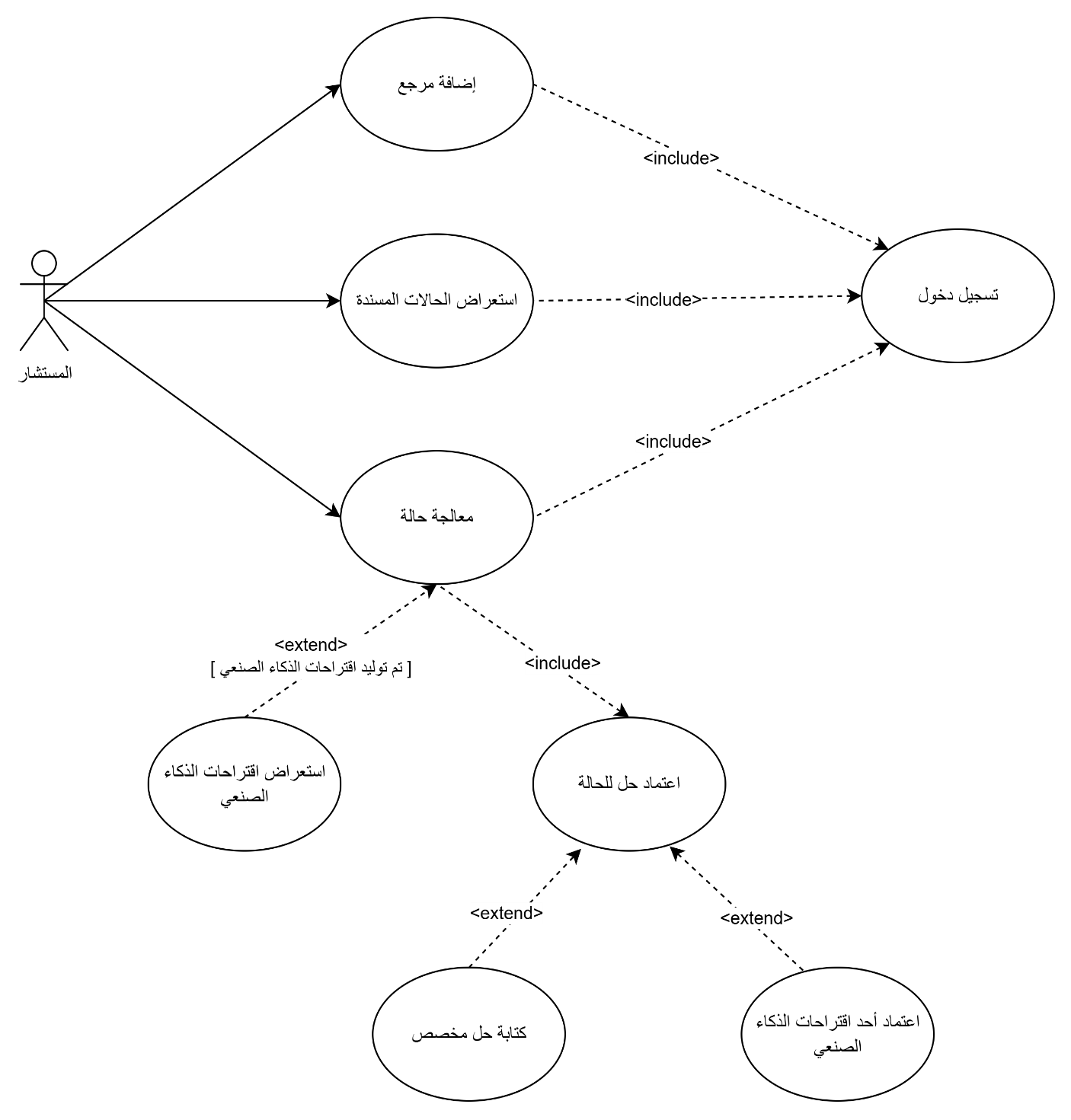
صورة 1: حالات استخدام إدارة حسابات المستخدمين



صورة 2: حالات استخدام مدير النظام



صورة 3: حالات استخدام المستشير



صورة 4: حالات استخدام المستشار

## السرد النصي لحالات الاستخدام

### إنشاء حساب جديد لمستشار

|  |  |
| --- | --- |
| اسم الحالة: إنشاء حساب جديد لمستشار | |
| الوصف Description | يقوم المستشار بإنشاء حساب جديد |
| الفاعلين Actors | المستشار |
| الشروط السابقة Precondition | لا يوجد |
| الشروط اللاحقة Postcondition | تم إنشاء حساب جديد للمستشار بحالة انتظار الموافقة من مدير النظام |

**سير الأحداث:**

**السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستشار |
|  | 1. يطلب إنشاء حساب جديد |
| 1. يطلب النظام المعلومات التالية:  * الاسم الأول * الاسم الأخير * الاختصاص * البريد الالكتروني * كلمة المرور * تأكيد كلمة المرور * الرقم الوطني |  |
|  | 1. يدخل المستخدم المعلومات المطلوبة ويطلب تأكيد العمليّة. |
| 1. يتحقق النظام من صحة المعلومات ويعيد للمستخدم رسالة توضح انتهاء العملية. |  |

**المسارات البديلة**

لا يوجد.

**مسارات الأخطاء**

**E1:** في المرحلة رقم 2 في حال تحقّقت إحدى الحالات التالية:

* البريد الالكتروني غير صالح.
* كلمة المرور ضعيفة.
* كلمة المرور لا تتطابق مع حقل تأكيد كلمة المرور.
* أحد الحقول لم يتم ملؤها.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الثانية في السيناريو الأساسي بالخطوة التالية:

2. يعيد النظام رسالة توضّح سبب الخطأ ويطلب تحديد المعلومات من جديد.

**E2:** في المرحلة رقم 4 في حال تحقّقت إحدى الحالات التالية:

* البريد الالكتروني مسجل مسبقاً ضمن النظام.
* الرقم الوطني مسجّل مسبقاً ضمن النظام.

في حال تحقّقت إحدى الحالات السابقة، يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

4. يعيد النظام رسالة توضّح سبب الخطأ ويطلب تحديد المعلومات من جديد.

### إضافة اختصاص

|  |  |
| --- | --- |
| اسم الحالة: إضافة اختصاص | |
| الوصف Description | يقوم مدير النظام بإنشاء مشروع جديد خاص به. |
| الفاعلين Actors | مدير النظام. |
| الشروط السابقة Precondition | مدير النظام مصادق مسبقاً لامتلاكه هذا الدور (دور مدير النظام). |
| الشروط اللاحقة Postcondition | تم إضافة اختصاص جديد لاختصاصات الاستشارات في النظام. |

**سير الأحداث**

**السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | مدير النظام |
|  | 1. يطلب إضافة اختصاص جديد. |
| 1. يطلب النظام اسم الاختصاص. |  |
|  | 1. يدخل المدير اسم الاختصاص. |
| 1. يتحقق النظام من اسم الاختصاص ويعيد للمدير رسالة توضح انتهاء العملية. |  |

**المسارات البديلة**

لا يوجد.

**مسارات الأخطاء**.

**E1:** في المرحلة رقم 4 في حال كان هناك اختصاص بنفس الاسم مسجّل مسبقاً ضمن النظام. يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

4. يعيد النظام رسالة توضّح أن هناك اختصاص موجود بنفس الاسم ويطلب تحديد الاسم من جديد.

### معالجة طلب إنشاء حساب لمستشار

|  |  |
| --- | --- |
| اسم الحالة: معالجة طلب إنشاء حساب لمستشار | |
| الوصف Description | يقوم مدير النظام بقبول أو رفض طلب مستشار. |
| الفاعلين Actors | مدير النظام. |
| الشروط السابقة Precondition | مدير النظام مصادق مسبقاً لامتلاكه هذا الدور (دور مدير النظام). |
| الشروط اللاحقة Postcondition | تم معالجة طلب إنشاء حساب لمستشار. |

**سير الأحداث**

**السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | مدير النظام |
|  | 1. يطلب عرض طلبات إنشاء الحساب لمستشارين. |
| 1. يقوم النظام بعرض الطلبات |  |
|  | 1. يختار عرض معلومات إحدى الطلبات. |
| 1. يقوم النظام بعرض تفاصيل الطلب. |  |
|  | 1. يختار مدير النظام أن يقبل أو يرفض الطلب. |
| 1. يعيد النظام للمستخدم رسالة توضح انتهاء العملية. |  |

**المسارات البديلة**

لا يوجد.

**مسارات الأخطاء**.

لا يوجد.

### طلب استشارة

|  |  |
| --- | --- |
| اسم الحالة: طلب استشارة | |
| الوصف Description | يقوم المستشير بملء استمارة لطلب استشارة. |
| الفاعلين Actors | المستشير. |
| الشروط السابقة Precondition | لا يوجد. |
| الشروط اللاحقة Postcondition | تم إرسال طلب الاستشارة ليتم معالجته. |

**سير الأحداث**

**السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | المستشير |
|  | 1. يطلب عملية طلب استشارة. |
| 1. يعرض النظام استمارة طلب الاستشارة تتضمن (البريد الالكتروني، اختصاص الاستشارة، موضوع الاستشارة، وصف الاستشارة) |  |
|  | 1. يملأ المستشير الاستمارة ويقوم بعملية الإرسال. |
| 1. يقوم النظام بإظهار رسالة إتمام العملية |  |

**المسارات البديلة**

لا يوجد.

**مسارات الأخطاء**.

لا يوجد.

### رفع مرجع

|  |  |
| --- | --- |
| اسم الحالة: رفع مرجع | |
| الوصف Description | يقوم المستشار بإضافة مرجع لمصادر المعرفة الخاصة به. |
| الفاعلين Actors | المستشار. |
| الشروط السابقة Precondition | المستشار مصادق مسبقاً لامتلاكه هذا الدور. |
| الشروط اللاحقة Postcondition | تم إضافة المرجع كمصدر للمعرفة. |

**سير الأحداث**

**السيناريو الأساسي الناجح**

|  |  |
| --- | --- |
| النظام | مدير النظام |
|  | 1. يطلب المستشار إضافة مرجع. |
| 1. يقوم النظام بطلب المرجع المراد رفعه. |  |
|  | 1. يقوم المستشار برفع المرجع. |
| 1. يقوم النظام بحفظ المرجع ضمن قاعدة المعطيات ويظهر رسالة بنجاح العملية. |  |

**المسارات البديلة**

لا يوجد.

**مسارات الأخطاء**.

**E1:** في المرحلة رقم 4 في حال تحقّقت إحدى الحالات التالية:

- في حال حدثت مشاكل في صيغة الملف المرفوع.

- في حال حدثت مشاكل في رفع الملف.

في حال تحققت إحدى الحالات السابقة، يتم استبدال الخطوة الرابعة في السيناريو الأساسي بالخطوة التالية:

4. يعيد النظام رسالة توضّح سبب الخطأ ويطلب رفع الملف من جديد.

# الفصل السادس

تصميم النظام

يعرض هذا الفصل القرارات التصميمية التي بني من خلالها النظام.

## مقدمة

تم اعتماد بنية الخدمات المصغّرة (Micro-Services) لبناء المنصة، يتكون النظام من مجموعة من الخدمات التي تعمل مع بعضها بشكل متكامل لخدمة المستخدم، تقدم هذه البنية مجموعة من الفوائد أهمها:

* **قابلية التوسع الأفقي (Horizontal Scalability):**

نستطيع زيادة عدد حاويات خدمة معينة عند ذروة الطلب بشكل مستقل دون التأثير في باقي مكونات النظام، الأمر الذي يتيح استخدام الموارد بكفاءة، إذ يتم حجز الموارد لأجزاء التطبيق التي تحتاجها فقط.

* **العزل ومقاومة الأعطال (Isolation & Fault Tolerance):**

لكل خدمة حاوية مستقلة تعمل ضمنها، إذا تعطلت إحدى الخدمات يتابع باقي النظام عمله، ويعاد تشغيل الجزء المتعطّل فقط، وبالتّالي تقل فترات التوقّف الكلّي.

* **التنوّع التقني:**

إن الفصل الواضح بين الخدمات يمكّننا من اختيار التقنية الأنسب لكل خدمة. مثل استخدام C#/ .NET لإدارة الحسابات والحالات، بينما اعتمدنا على Python/FastAPI في خدمات الذكاء الصنعي والتضمين.

* **سرعة النشر المستقل:**

يمكن إصدار نسخة جديدة من أي خدمة – سواء لإصلاح عطل أو لإضافة ميزة – من دون إعادة بناء أو نشر النظام بأكمله، ما يقلّل زمن التوقّف، ويقوم بتسريع دورة التطوير (CI/CD).

* **سياق محدود (Bounded Context):**

تم تقسيم المنصّة إلى سياقات عمل واضحة حسب المهام، الأمر الذي قلّل التبعيات وجعل تطوير كل سياق واختباره وفهمه أكثر بساطة.

نعرض في الفقرات التالية شرحاً لتصميم كل خدمة.

## خدمة البوابة Gateway Service

تشكّل البوابة المدخل الوحيد (Single Entry Point) إلى المنصّة، وتعمل وفق نمط (Reverse Proxy).

وظائفها الرئيسية هي:

1. استقبال الطلبات الخارجيّة من الواجهة الأمامية أو أي عميل آخر.
2. التحقق من الهويّة والصلاحيات قبل توجيه الطلب.
3. موازنة الحمل (Load Balancing) بين نسخ الخدمات المتعددة لضمان استجابة متّسقة وتوزيع الضغط.

أهميتها:

1. فصل الاهتمامات: حيث تعفى كل خدمة من تحمل عبء المصادقة وموازنة الحمل، فتركّز على مهامها ومنطقها الخاص.
2. سهولة التوسّع: إضافة نسخة جديدة من أي خدمة تتم بسهولة، حيث أن البوّابة تضبط التوجّه تلقائيّاً.
3. تعزيز الأمان: وجود البوّابة يمنع كشف تفاصيل البنى الداخلية.

سلبيّاتها:

1. قد تتحوّل البوابة إلى نقطة اختناق إذا لم تُنشر بنسخ احتياطيّة أو خلف موزّع حمل أعلى.
2. وجود البوابة يعني إضافة قفزة شبكيّة بين العميل والخدمة، الأمر الذي بسبب ارتفاعاً طفيفاً في زمن الاستجابة لكنّه مقبول مقارنة بما يوفّره من أمان.

## خدمة الحالة Case Service

تُعدّ خدمة الحالة مركز دورة حياة الاستشارة؛ فهي تستقبل طلب الاستشارة الوارد وتقوم بما يلي:

* تنشئ «حالة استشارية» جديدة وتضيفها إلى قاعدة البيانات.
* تتابع انتقال الحالة بين المراحل الأربعة (جديد، بانتظار الذكاء الصنعي، بانتظار مراجعة المستشار، منته).
* تصدر إشعارات إلى الخدمات الأخرى عند تغيّر الحالة.
* تضيف اقتراحات الذكاء الصنعي عند توليدها.
* وتحفظ الحلّ المعتمد عن تسلّمه.

المعمارية المعتمدة البنية النظيفة (Clean Architecture) والتي تتكون من الطبقات التالية:

1. طبقة النطاق (Domain): تُعرّف المفاهيم والكيانات الأساسية للنظام بمعزل تامّ عن أي تفاصيل تقنية.
2. طبقة التطبيق (Application): هي المسؤولة عن حالات الاستخدام حيث تربط طلبات العملاء بكيانات طبقة النطاقة دون الانشغال بآليات التخزين أو الاتّصال.
3. طبقة البنية التحتيّة (Infrastructure): توفّر الوسائل التقنية التي تحتاجها الخدمة – كقواعد البيانات وأنظمة الرسائل والتكاملات الخارجية – بقابليّة استبدال سهلة.
4. طبقة العرض (Presentation): تقدّم واجهات التفاعل، سواء كانت نقاط نهاية (API Endpoint) أو مستهلكي أحداث (Consumers)، وتقوم بتحويل كل طلب وارد إلى طبقة التطبيق.

تم اختيار هذه البنية وذلك للأسباب التالية:

1. قابلية اختبار مرتفعة: يمكن اختبار منطق العمل (Business Logic) بمعزل عن قاعدة البيانات ورتل الرسائل.
2. مرونة استبدال التقنيات: فمثلاً تغيير قاعدة البيانات أو رتل الرسائل المستخدم لا يمسّ سوى طبقة البنية التحتيّة.
3. إعادة استخدام منطق الأعمال: يمكن استخدام بعض حالات الاستخدام من واجهة REST أو من مستهلك الأحداث دون تكرار الكود.

إن اعتماد هذه البنية تطلّب جهداً تمهيديّاً ووقتاً أطول في الإعداد، لكن هذا الاستثمار المبدئيّ عوّض لاحقاً بسهولة الصيانة والتوسّع.

## خدمة المستشار (Consultant Service)

تتولّى هذه الخدمة إدارة دورة حياة المستشارين بالكامل:

* إنشاء حساب المستشار وربطه بخدمة المصادقة.
* تحديث بياناته وحالته من (بانتظار الموافقة إلى مقبول أو مرفوض).
* مراقبة المستشارين وعدد الحالات المفتوحة لديه (بانتظار المراجعة).
* تعيين الحالات الواردة إلى المستشار الأنسب وفق اختصاصه وتوافره بالإضافة إلى بعض المعلومات الأخرى.
* بثّ الأحداث الرئيسية إلى رتل التراسل تماماً كما تقوم به خدمة الحالة، لإبقاء الخدمات الأخرى متزامنة.

إنّ المنطق الكامن وراء اختيار المستشار ومتابعة الحالات الموكلة إليه معقّد؛ إذ يجب موازنة عوامل متشابكة—تخصّص المستشار، وعدد الحالات المفتوحة لديه وغيرها من العوامل التي قد تعطي المستشار أولويّة أكبر في تعيين الحالة له (عدد الحالات التي قام بمعالجتها ضمن فترة زمنية محددة) – مع الحفاظ على اتّساق البيانات عند تنافس النسخ المختلفة للخدمة على تعيين الحالات.

تم تطوير هذه الخدمة أيضاً وفق معمارية البنية النظيفة لعزل منطق العمل عن تفاصيل التخزين أو المراسلة، مع إبقائها قابلة للتوسّع والصيانة المستمرّة؛ إذ يُخطَّط لإضافة تحسينات دوريّة - مثل خوارزميات ترشيح أدقّ لتعيين المستشار، ومقاييس زمنية لتتبّع أعبائه، ومعايير لتقييم المستشار - حتى تظلّ متماشية مع نموّ المنصّة ومتطلّباتها المستقبلية.

## خدمة المصادقة (Auth Service)

تمثّل خدمة المصادقة بوّابة هويّة المنصّة؛ والوسيلة التي تستخدمها البوابة لإجراء عملية التفويض وتقوم بالمهام التالية:

* تُنشى حسابات المستخدمين (تحفظ البريد الالكتروني وتشفير كلمة السر فقط).
* ترسل طلباً لخدمة المستشار لإضافة مستشار جديد.
* تقوم بتوليد رموز JWT اللازمة للدخول الآمن والتحقق منها عند طلب خدمة البوّابة ذلك.
* تزوّد خدمة البوّابة بالمعلومات الضرورية مثل (مثل المعرّف الخاص بالمستخدم ودور المستخدم) كي تنجز عملية التفويض.
* تقوم بتوليد Hash لكلمة السر قبل التخزين.

تم اعتماد المعمارية 3-tier في بناء هذه الخدمة وتتكون هذه الطبقة من:

1. طبقة العرض (Presentation Layer): وتحتوي على نقاط نهاية REST Endpoints لاستقبال طلبات التسجيل/ الدخول والتحقق من JWT.
2. طبقة المنطق (Service Layer): تطبّق قواعد التحقق (بريد الكتروني فريد، شروط كلمة السر)، كما تستدعي منشئ الرموز أو مولّد ال Hash.
3. طبقة البيانات (Data Access Layer): تعامل مباشر مع قاعدة البيانات الخاصة بالمستخدمين.

استخدمنا هذه المعمارية للأسباب التالية:

1. منطق العمل محدود: لا حاجة لبنية أعمق متل البنية النظيفة.
2. سهولة القراءة والصيانة: تقسيم المكوّنات إلى طبقات مستقلّة يجعل بنية الكود سهلة الفهم ويُمكّن المطورين من تنفيذ التعديلات أو إصلاح الأخطاء بسرعة دون التوغّل في تفاصيل غير معنيّة.
3. قابليّة التوسّع الأفقي: نستطيع تشغيل نسخ إضافيّة من طبقتَي العرض والمنطق خلف موازِن حمل واحد، مع الإبقاء على قاعدة بيانات موحَّدة تتشاركها تلك النسخ.

إن القرار بالسماح لخدمة البوابة بالاعتماد على خدمة المصادقة لفكّ رموز JWT يضيف قفزة شبكيّة إضافيّة، ومع تزايد الحمل قد يتطلّب الأمر ذاكرة خبيئة أو تحسين لمسار الطلب. ومع ذلك يبقى الثمن مقبولاً مقابل مركزيّة الأمان التي توفرها خدمة المصادقة.

## خدمة التضمين Embedding Service

### تمهيد

تعد هذه الخدمة هي حلقة الوصل بين مصادر المعرفة (مراجع وحالات سابقة) وخدمة الذكاء الصنعي، إذ تقوم بتضمين مصادر المعرفة (توليد متّجه التمثيل Vector Representation) وتخزينها وتنفيذ عمليّات البحث بالتشابه عليها (Similarity Search).

### مهام الخدمة

1. تقسيم المراجع إلى قطع نصّيّة صغيرة (Chunks) مع تنفيذ تنظيف مسبق للنص.
2. توليد شعاع تضمين لكل قطعة نصيّة لمرجع أو لحالة تم معالجتها من قبل مستشار.
3. تخزين أشعة التضمين في قاعدة البيانات.
4. تزويد خدمة الذكاء الصنعي بنقاط نهاية لإجراء البحث بالتشابه وإيجاد مصادر المعرفة المتعلقة بسياق الحالة التي تقوم الخدمة باقتراح حلول لها.
5. استقبال حدث انتهاء معالجة حالة من خدمة الحالة لتضمين الحالة مع الحل المعتمد وإضافتها إلى مصادر المعرفة.

### المعمارية المعتمدة

تم اعتماد معمارية البنية النظيفة وذلك للأسباب التالية:

1. فصل الاهتمامات يسمح باختبار خوارزميات معالجة النصوص أو خوارزميات التضمين بمعزل عن قاعدة البيانات، أو اختبار الطريقة المستخدمة لإجراء البحث بالتشابه بمعزل عن غيرها.
2. سهولة استبدال نموذج التضمين مستقبلاً من دون تعديل طبقة الواجهة.

### قاعدة البيانات

تم الاعتماد على pgvector داخل PostgreSQL، ليبقى النظام مبنيّاً بشكل كامل على قاعدة SQL موحّدة من جهة، ويستفيد في الوقت نفسه من إمكانات الفهرسة المتجهيّة عالية الأداء من جهة أخرى.

يمنحنا pgvector فهرس HNSW مدمجًا مباشرة في المحرّك، فيتيح تنفيذ استعلامات «أقرب K متجه» بزمنٍ قصير حتى مع ملايين السجلات. والأهم أنّ المتجهات تُخزَّن وتُسترجَع عبر أوامر SQL نفسها، ما يوفّر **اندماجًا كاملًا** بين البحث الدلالي والعمليات العلائقيّة التقليدية (مثل الفلترة حسب التخصّص أو معرّف المستشار) في استعلام واحد، ويحافظ على مصدر بيانات موحَّد يسهل تشغيله وصيانته.

تعتمد قاعدة البيانات في خدمة التضمين على ثلاث جداول رئيسيّة:

1. جدول الحالات المعالَجة: يخزّن وصف الحالات التي جرى معالجتها مسبقاً، بالإضافة إلى شعاع التضمين الخاص بكل حالة، بهدف استخدامها كمصادر معرفة مستقبلاً.
2. جدول المراجع: يحتفظ بمعلومات عامّة عن كل مرجع مرفوع من قبل المستشار مثل العنوان وحالة المعالجة (يتم استخدام هذه المواصفة لمتابعة عمليّة رفع المقاطع النصيّة للمرجع).
3. جدول المقاطع: يُقسَّم كل مرجع إلى عدة مقاطع نصيّة، ويُحفظ كل مقطع مع شعاع التضمين المرتبط به، مع وجود مرجع إلى المصدر الأصلي.

### الأنماط التصميمية المستخدمة

**لا تنسى أن تضيفهم هنا**

## خدمة الذكاء الصنعي

### تمهيد

تتولى هذه الخدمة توليد مقترحات الحلول لأي حالة استشارية فور تعيين مستشار لها، مستندةً إلى مصادر المعرفة الخاصة بذلك المستشار.

### مهام الخدمة

تقوم خدمة الذكاء الصنعي بالمهام التالية:

1. الاستماع لحدث تعيين حالة لمستشير الصادر عن خدمة المستشير.
2. تنظيف وصف الحالة وتنفيذ ترجمة، تلخيص أو إعادة صياغة له عند الحاجة ليصبح جاهزاً للتضمين.
3. استرجاع مصادر المعرفة عن طريق الطلب من خدمة التضمين للحصول على المقاطع أو الحالات الأكثر صلة بالوصف.
4. تشكيل أمر Prompt يدمج وصف الحالة ومصادر المعرفة، ومن ثم استدعاء نموذج لغة كبير LLM لإنتاج مجموعة حلول مرتّبة.
5. نشر حدث الانتهاء من اقتراح حلول للحالة عبر رتل الرسائل، لتستمع له خدمة الحالة.

### المعمارية المعتمدة

تم اعتماد معمارية البنية النظيفة وذلك للأسباب التالية:

1. طبقات النطاق والتطبيق تحوي منطق إعداد الأمر واستخراج ومعالجة المقترحات من الإجابة.
2. طبقة البنية التحتيّة تضم تنفيذ استدعاء LLM وطلب مصادر المعرفة من خدمة التضمين.
3. يمكن تعديل طريقة بناء الأمر دون المساس بطبقة العرض.
4. يمكن تبديل LLM محلّي بنموذج خارجي أو العكس دون المساس بطبقة التطبيق أو النطاق.
5. يمكن اختبار نموذج LLM بشكل منفصل وتقييم نتائجه دون الحاجة لتشغيل الخدمة بشكل كامل.

### الأنماط التصميمية المستخدمة

**لا تنسى أن تضيفهم هنا**

لا تحتاج هذه الخدمة لأي قاعدة بيانات فهي عديمة الحالة (stateless)، وتكتفي بمعالجة الطلب لحظيّاً ثم نشر الحدث دون تخزين دائما للبيانات.