# تقرير فني: المعمارية الكاملة ونشر مشروع "مرشدي الأكاديمي الذكي"

## مقدمة: ترجمة الرؤية إلى معمارية قابلة للتنفيذ

يهدف هذا التقرير إلى تقديم مخطط فني قابل للنشر (Technical Blueprint) لمشروع "مرشدي الأكاديمي الذكي". التحول من مفهوم مبتكر إلى نظام برمجي فعال ومستدام يتطلب قرارات معمارية دقيقة. سيقدم هذا التحليل خارطة طريق تقنية شاملة تركز على الكفاءة، وقابلية التوسع، وآلية تنفيذ الميزات الثلاثية التي تم تحديدها: الأساسية (الاسترجاع المعزز)، السوبر المميزة (التنبؤ)، وفائقة الذاء (التحليل المستند إلى الرسوم البيانية).

لتلبية هذه المتطلبات المتنوعة، يُقترح مكدس تقني (Tech Stack) حديث ومترابط، مصمم خصيصاً لتطبيقات الذكاء الاصطناعي التوليدي المعقدة. يتكون هذا المكدس من خمسة مكونات رئيسية تعمل بتناغم ضمن بيئة مُحَوّاة (Containerized):

1. **الواجهة الأمامية (Frontend):** Streamlit
2. **الواجهة الخلفية (Backend):** FastAPI
3. **خدمة النماذج اللغوية (LLM Serving):** Ollama (للتطوير والنشر الأولي)
4. **قاعدة بيانات المتجهات (Vector DB):** ChromaDB
5. **قاعدة بيانات الرسم البياني (Graph DB):** Neo4j

سيفصل هذا التقرير كيفية دمج هذه المكونات، وتكوينها، ونشرها بفعالية لتحقيق نظام إرشاد أكاديمي ذكي وقوي.

## القسم 1: الأسس المعمارية ومبادئ النشر الفعّال

### 1.1. التحليل المعماري: لماذا يعتبر Docker ضرورة حتمية لمشروعك (وليس مجرد خيار)

الإجابة المباشرة على استفسار التشغيل هي: نعم، مشروع "مرشدي الأكاديمي الذكي" يتطلب استخدام Docker بشكل قاطع لضمان نجاحه وقابليته للتطوير والنشر.

الاختلاف الجوهري بين بيئة افتراضية (مثل venv) وحاوية Docker يكمن في مستوى العزل. البيئة الافتراضية venv تقوم فقط بعزل تبعيات Python.1 في المقابل، توفر حاوية Docker عزلاً كاملاً لنظام التشغيل بأكمله.2 الحاوية تتضمن نظام الملفات الخاص بها، وهيكل التبعيات النظامية (مثل مكتبات C++ أو CUDA)، والعمليات، وإعدادات الشبكة.2

تطبيقات الذكاء الاصطناعي، وخاصة تلك التي تدمج نماذج لغوية (LLMs)، ونماذج تنبؤية، وأدوات معالجة لغات، هي عرضة لما يُعرف بـ "جحيم التبعيات" (Dependency Hell). على سبيل المثال:

* قد يتطلب النموذج اللغوي (LLM) إصداراً محدداً من torch و cuda-12.1.
* قد تتطلب مكتبة التنبؤ (مثل scikit-learn 4) إصداراً مختلفاً من numpy عن ذلك الذي يتطلبه torch.
* قد تحتاج مكتبة معالجة اللغات (مثل spacy 5) إلى مكتبات نظام (build-essential 6) لا علاقة لها بالمكونات الأخرى.

البيئة الافتراضية venv عاجزة تماماً عن حل تعارضات مكتبات النظام (System Libraries) مثل CUDA. يوفر Docker حلاً لهذه المشكلة من خلال ضمان "تسليم سريع ومتسق للتطبيقات" 7، حيث يتم تغليف التطبيق مع *جميع* متطلباته داخل حاوية واحدة يمكن تشغيلها "على أي نظام تشغيل" 2، بدءاً من جهاز المطور المحلي وصولاً إلى أي مزود سحابي.7 هذا يضمن أن النظام الذي يعمل في بيئة التطوير سيعمل تماماً بنفس الطريقة في بيئة الإنتاج.

### 1.2. المعمارية المقترحة: تصميم النظام المنفصل (Decoupled Architecture)

لتحقيق الكفاءة والاستجابة المطلوبة، يُقترح بشدة اعتماد معمارية منفصلة (Decoupled Architecture) تفصل تماماً الواجهة الأمامية (Frontend) عن الخدمات الخلفية (Backend).

* الواجهة الأمامية (Frontend): Streamlit  
  يُعد Streamlit الخيار الأمثل لبناء الواجهة، كونه "الطريقة الأسرع لبناء ومشاركة تطبيقات البيانات" 8 ويوفر مكونات جاهزة لبناء واجهات الدردشة التفاعلية.8
* الواجهة الخلفية (Backend): FastAPI  
  يعمل FastAPI كـ "إطار ويب حديث وعالي الأداء".9 نظراً لدعمه الأصيل للبرمجة غير المتزامنة (async) 11، فهو مثالي ليكون "الجسر" (Bridge) الذي يربط بين واجهة المستخدم ونماذج تعلم الآلة الثقيلة.9

يكمن السبب الحاسم وراء هذا الفصل في طبيعة عمل Streamlit وميزات المشروع. يعمل Streamlit بنموذج "إعادة التشغيل عند التفاعل" (rerun-on-interaction)، حيث "يعيد تشغيل البرنامج النصي بأكمله" مع كل تفاعل للمستخدم (مثل النقر على زر).13 ميزاتك "السوبر" و "فائقة الذكاء" (مثل استعلام RAG أو تشغيل نموذج التنبؤ) هي عمليات بطيئة قد تستغرق عدة ثوانٍ.

إذا تم دمج هذا المنطق الثقيل مباشرة في برنامج Streamlit النصي، فإن واجهة المستخدم "ستتجمد" (Freeze) بالكامل في كل مرة يطرح فيها الطالب سؤالاً، مما يؤدي إلى تجربة مستخدم سيئة للغاية.

الحل المعماري 9 هو أن تعمل واجهة Streamlit كـ "واجهة طرفية بسيطة" (Dumb Terminal). عند إرسال السؤال، كل ما تفعله هو استخدام مكتبة requests 11 لإرسال طلب HTTP إلى نقطة نهاية (endpoint) محددة في FastAPI.16 يتلقى FastAPI (الذي يعمل كخدمة منفصلة) الطلب، ويدير المهمة الحسابية الثقيلة في الخلفية (استدعاء النموذج اللغوي، استعلام قاعدة البيانات)، ثم يعيد النتيجة. هذا التصميم يحرر واجهة Streamlit لتبقى سريعة الاستجابة، مما يسمح لها بعرض مؤشرات الانتظار (مثل st.status 17) بينما تتم معالجة الطلب في الخلفية.

### 1.3. استراتيجية إدارة الحالة (State Management) والتفاعلات

أحد التحديات الرئيسية في تطبيقات Streamlit هو إدارة الحالة (State)، خاصة في تطبيقات الدردشة التي تحتاج إلى "تذكر" سياق المحادثة.

يوفر Streamlit آلية مدمجة تسمى st.session\_state.18 هذه الآلية هي "طريقة لمشاركة المتغيرات بين عمليات إعادة التشغيل، لكل جلسة مستخدم" 18، وهي ضرورية لبناء تطبيقات المحادثة 17 لتخزين سجل الرسائل المعروضة حالياً.

ومع ذلك، فإن st.session\_state هي حالة سريعة الزوال (Ephemeral)؛ أي أنها تُفقد بمجرد إغلاق المستخدم لعلامة تبويب المتصفح. المرشد الأكاديمي الذكي يحتاج إلى ذاكرة طويلة الأمد.

لذلك، يجب اعتماد "استراتيجية إدارة حالة هجينة ذات مستويين" (Hybrid 2-Tier State Management):

1. المستوى 1: حالة واجهة المستخدم السريعة (Ephemeral UI State)  
   تُستخدم st.session\_state 18 حصرياً لتخزين سجل الدردشة المعروض حالياً على الشاشة. عندما يكتب المستخدم رسالة، يتم إضافتها فوراً إلى st.session\_state['messages'] ويتم إعادة عرض واجهة المستخدم على الفور لإظهار الرسالة الجديدة. هذا يوفر استجابة فورية للمستخدم.
2. المستوى 2: حالة التطبيق الدائمة (Persistent Application State)  
   بعد تحديث الحالة المحلية (المستوى 1)، يقوم Streamlit بإرسال الرسالة إلى الواجهة الخلفية (FastAPI).11 الواجهة الخلفية (FastAPI) تقوم بتحميل سجل المحادثة الدائم والكامل للطالب من قاعدة بيانات علائقية (مثل Postgres أو SQLite)، تضيف الرسالة الجديدة، تستدعي النموذج اللغوي للحصول على الرد، ثم تحفظ كلاً من رسالة المستخدم ورد الذكاء الاصطناعي مرة أخرى في قاعدة البيانات الدائمة قبل إرسال الرد إلى Streamlit.20

هذا النمط الهجين يمنح النظام أفضل ما في العالمين: الاستجابة الفورية لواجهة المستخدم التي يوفرها st.session\_state 18، مع ضمان الاستمرارية والذاكرة الدائمة التي توفرها الواجهة الخلفية FastAPI وقاعدة البيانات.20

## القسم 2: المكدس التقني الأساسي (Core Stack) لتشغيل النماذج اللغوية محلياً

يتطلب تشغيل "المرشد الذكي" بكفاءة اختيار إطار عمل لخدمة النماذج اللغوية (LLM Serving Framework)، وفهم تقنيات التحسين (Quantization)، وتحديد متطلبات الأجهزة (Hardware).

### 2.1. اختيار خادم النموذج اللغوي المحلي (LLM Serving Framework)

هناك عدة خيارات لخدمة النماذج اللغوية محلياً، ولكل منها استخدامه الأمثل.

* **Ollama:** يُعتبر "مثالياً للتطوير المحلي والنماذج الأولية".21 قوته تكمن في "بساطته" وسهولة استخدامه.21
* **vLLM:** يوصف بأنه "متفوق بشكل لا لبس فيه للنشر الإنتاجي".21 إنه مصمم "للأداء العالي" (High-Throughput) 24 ويتفوق بشكل كبير على Ollama في التعامل مع الأحمال المتزامنة.21
* **Llama.cpp:** هو الأساس الذي بُني عليه الكثير، وهو "مناسب لوحدة المعالجة المركزية (CPU)" 23 وغالباً ما يُعتبر أداؤه "أفضل من ollama و transformers" في سيناريوهات معينة.26 كما أن Docker يوفر الآن "Docker Model Runner" الذي يستخدم llama.cpp.27

الاختيار هنا ليس "إما/أو" بل هو قرار يعتمد على المرحلة. المسار الأكثر كفاءة هو اعتماد "استراتيجية أدوات مرحلية":

1. **مرحلة التطوير (Development):** ابدأ باستخدام **Ollama** مباشرة على جهازك المحلي.23 هذا يتيح لك بدء البرمجة والتجربة فوراً.
2. **مرحلة النشر (Deployment):** عندما تنتقل إلى بيئة Docker، استخدم **Ollama** كـ "خدمة" رسمية (باستخدام image: ollama/ollama) داخل ملف docker-compose.yml.29 هذا يمنحك بساطة Ollama ضمن بيئة Docker المعزولة والمتسقة.
3. **مرحلة الإنتاج (Production Scaling):** إذا واجه المشروع ضغطاً عالياً وتطلب أداءً فائقاً (tokens/sec)، عندها فقط يتم استبدال خدمة Ollama بخدمة **vLLM**.21 هذا يتطلب تكويناً أكثر تعقيداً ويركز بشدة على وحدات معالجة الرسومات من NVIDIA.22

للمشروع الحالي، التوصية هي البدء والالتزام بـ Ollama (المرحلة 1 و 2) لضمان التوازن الأمثل بين سهولة الاستخدام والأداء الكافي.

**جدول 1: مقارنة خوادم النماذج اللغوية المحلية (LLM Serving Frameworks)**

| **الإطار (Framework)** | **حالة الاستخدام الأساسية** | **الأداء (TPS مقابل سهولة الاستخدام)** | **دعم الأجهزة (Hardware Support)** |
| --- | --- | --- | --- |
| **Ollama** | تطوير ونماذج أولية سريعة 21 | سهل الاستخدام جداً، أداء أبطأ تحت الضغط 21 | ممتاز: NVIDIA GPU, Apple Silicon (Mac), CPU 23 |
| **vLLM** | نشر إنتاجي عالي الأداء 21 | أداء (TPS) هو الأعلى، إعداد معقد 24 | يركز بشدة على NVIDIA GPUs 22 |
| **Llama.cpp** | مرن، أساس تقني 23 | أداء ممتاز، خاصة على CPU 23 | مرن جداً: NVIDIA, Apple, CPU 26 |

### 2.2. تقنيات التحسين (Quantization): GGUF مقابل GPTQ

النماذج اللغوية الحديثة ضخمة جداً. تشغيلها محلياً يتطلب "تحسيناً" (Quantization)، وهي عملية تقليل دقة أوزان النموذج لتقليل حجمه ومتطلبات الذاكرة.

* **GPTQ (Generative Pre-trained Transformer Quantization):** هي تقنية تركز بشكل أساسي على "استدلال GPU".31 إنها "متوافقة مع معظم أجهزة GPU" (تحديداً NVIDIA) 31 وتتطلب وجود GPU لتشغيلها بفعالية.32
* **GGUF (GGML Universal Format):** هو "تنسيق ملف عالمي" 33 مصمم خصيصاً "للتشغيل عبر المنصات" (Cross-Platform Deployment).33 على عكس GPTQ، يركز GGUF على العمل على CPU، وأجهزة Apple M، ويوفر "مرونة في تفريغ (offloading) الطبقات إلى GPU" حسب الحاجة.31

بالنسبة لمشروع يهدف إلى "الكفاءة" ويعتمد على Docker (الذي يهدف إلى "التشغيل في أي مكان" 7)، فإن **GGUF هو الخيار الوحيد المنطقي**.

السبب هو المرونة. GPTQ يقيدك ببيئات NVIDIA GPU فقط 32، مما يتعارض مع مبدأ "التشغيل في أي مكان" لـ Docker. GGUF 33، من ناحية أخرى، مصمم للعمل على أي جهاز:

* على خادم إنتاجي ببطاقة NVIDIA، سيقوم GGUF (عبر Ollama/Llama.cpp) بتفريغ 100% من النموذج إلى الـ GPU.
* على جهاز ماك M1/M2 للمطور، سيعمل بكفاءة على Apple Silicon.31
* على خادم CPU (للاختبارات)، سيظل يعمل (وإن كان ببطء).

لضمان أقصى قدر من قابلية النقل والكفاءة، يجب توحيد جميع النماذج اللغوية المستخدمة في المشروع لتكون بتنسيق GGUF (بتحسين 4-bit أو 5-bit).

### 2.3. متطلبات الأجهزة (VRAM) للنماذج المحلية

حجم ذاكرة الوصول العشوائي للفيديو (VRAM) في بطاقة الرسومات (GPU) هو العامل المحدد الأكبر لتشغيل النماذج اللغوية محلياً.

* **بدون تحسين:** تشغيل نموذج Llama 3 8B (8 مليار بارامتر) يتطلب حوالي 18.4GB VRAM.34 نموذج 7B يتطلب حوالي 28GB VRAM (7 \* 4 bytes).35 هذه الأرقام تجعل تشغيلها على الأجهزة الاستهلاكية شبه مستحيل.
* **مع تحسين GGUF (4-bit):** تنخفض المتطلبات بشكل كبير. نموذج Llama 3 8B (4-bit) يحتاج فقط إلى 5 - 5.73GB VRAM.36 نموذج Mistral 7B (4-bit) يعمل بشكل مريح على بطاقة 8GB VRAM.37

ومع ذلك، ذاكرة النموذج ليست كل شيء. يجب أيضاً حساب الذاكرة المطلوبة لـ "السياق" (Context Window)، وهو حجم النص (الأسئلة + مستندات RAG) الذي يمكن للنموذج "رؤيته" في المرة الواحدة. تشير التقديرات إلى الحاجة لـ "2GB إضافية للسياق" 36 لحجم 8k.

هنا تظهر "النقطة المثلى" (sweet spot).

* **بطاقة 8GB VRAM** (مثل RTX 4060 أو RX 6600 37): هي **الحد الأدنى** العملي. ستكون كافية لتشغيل نموذج 7B أو 8B 37، لكنها ستجبرك على استخدام سياق صغير (مثل 4k أو 8k).36 هذا يحد من فعالية RAG، حيث لا يمكنك إرسال الكثير من المستندات المسترجعة إلى النموذج.
* **بطاقة 12GB VRAM** (مثل RTX 3060 12GB 36): هي **التوصية القوية**. هذه الذاكرة الإضافية تتيح لك تشغيل نفس نموذج 8B ولكن مع سياق ضخم (مثل 32k). أشار أحد المستخدمين إلى أن سياق 32k يمكن أن يستهلك ما يصل إلى 11GB VRAM.36 هذا السياق الأكبر يعني أن "المرشد الذكي" يمكنه تحليل مستندات أكاديمية أطول وأكثر تعقيداً، مما يحسن جودة الإجابات بشكل كبير.

**جدول 2: متطلبات VRAM التقديرية (لنماذج GGUF 4-bit)**

| **النموذج (Model)** | **VRAM الأساسي (النموذج فقط)** | **VRAM المقدر (مع سياق 8k)** | **VRAM المقدر (مع سياق 32k)** | **الحد الأدنى للبطاقة الموصى بها** |
| --- | --- | --- | --- | --- |
| **Mistral 7B** | ~4.1 GB 37 | ~6.1 GB | ~9-10 GB | 8 GB (للسياق الأساسي) |
| **Llama 3 8B** | ~5.7 GB 36 | ~7.7 GB | ~11-12 GB 36 | 12 GB (للسياق الكامل) |

## القسم 3: تنفيذ الميزات الأساسية: نظام الاسترجاع المعزز (RAG) للمعرفة الأكاديمية

تمثل ميزاتك "الأساسية" (مثل الإجابة على الأسئلة من لوائح الجامعة أو المناهج الدراسية) تطبيقاً كلاسيكياً لـ "الاسترجاع المعزز بالتوليد" (Retrieval-Augmented Generation - RAG).38

### 3.1. اختيار إطار التنسيق (Orchestration Framework): LangChain مقابل LlamaIndex

لبناء نظام RAG، تحتاج إلى إطار عمل "لتنسيق" (Orchestrate) العملية (تحميل المستندات، تقطيعها، تضمينها، استرجاعها، ثم تمريرها إلى النموذج اللغوي).

* **LlamaIndex:** هو إطار عمل "مثالي إذا كان تركيزك الأساسي على فهرسة واسترجاع البيانات بكفاءة".39 وقد أشار مستخدمون إلى أنه "أفضل بكثير لعمليات الاستيعاب والتخزين والاسترجاع".41
* **LangChain:** هو "إطار قوي لبناء تطبيقات LLM شاملة" 39، بما في ذلك RAG.42

على الرغم من أن LlamaIndex قد يكون متفوقاً في مهمة RAG *النقية* 41، إلا أن مشروعك *ليس* مشروع RAG نقي. إنه نظام هجين معقد يدمج RAG (الميزات الأساسية)، ونماذج Scikit-learn (الميزات السوبر)، وقواعد بيانات الرسم البياني (الميزات فائقة الذكاء).

هذا التعقيد يتطلب "منسقاً" (Orchestrator) يمكنه *توجيه* (Route) سؤال المستخدم بذكاء إلى الأداة الصحيحة.

* سؤال "ما هي متطلبات هذه المادة؟" يجب أن يذهب إلى نظام RAG (القسم 3).
* سؤال "هل أنا في خطر الرسوب؟" يجب أن يذهب إلى نموذج Scikit-learn (القسم 4).
* سؤال "ما المسار الدراسي الذي يوصلني لتخصص الذكاء الاصطناعي؟" يجب أن يذهب إلى نظام Graph RAG (القسم 5).

**LangChain** 42 مصمم خصيصاً ليكون هذا "الغراء" (Glue) أو المنسق الشامل. بينما LlamaIndex 39 هو أداة متخصصة، فإن LangChain هو صندوق الأدوات. لهذا المشروع المعقد، التوصية هي استخدام **LangChain** كنظام تنسيق أساسي.

### 3.2. استراتيجيات التقطيع (Chunking) المتقدمة للمستندات الأكاديمية (PDFs)

جودة نظام RAG تعتمد بشكل شبه كامل على استراتيجية "التقطيع" (Chunking).43 المستندات الأكاديمية (اللوائح، المناهج الدراسية، الأوراق البحثية) 44 هي مستندات منظمة هيكلياً.

استخدام استراتيجيات بسيطة مثل "التقطيع ثابت الحجم" (Fixed-Size Chunking) هو "اعتباطي" 43 وسيفشل حتماً، لأنه قد يفصل جدولاً عن عنوانه، أو يقطع جملة في منتصفها. "التقطيع الدلالي" (Semantic Chunking) 46 أفضل، لكنه قد يدمج أقساماً مختلفة (مثل "وصف المساق" و "المتطلبات المسبقة") لأنها متشابهة دلالياً، وهو ما لا نريده.

الحل هو "استراتيجية تقطيع هرمية" (Hierarchical Chunking Strategy)، والمعروفة أيضاً بنمط "الآباء والأبناء" (Parent-Child Retriever) 47:

1. **الخطوة 1: التقطيع الهيكلي (الأب):** أولاً، يتم استخدام أداة تحليل للمستندات (مثل pdfplumber 48 أو unstructured.io 48) لتنفيذ "تقطيع مستند إلى المستند" (Document-Based Chunking).46 يتم تقسيم ملف PDF إلى "قطع أصل" (Parent Chunks) كبيرة بناءً على هيكله، مثل العناوين الرئيسية (e.g., "القسم 1: المتطلبات المسبقة"، "القسم 2: مخرجات التعلم").
2. **الخطوة 2: التقطيع المتكرر (الابن):** ثانياً، *داخل* كل "قطعة أصل"، يتم استخدام "التقطيع المتكرر" (Recursive Chunking) 50 لتقسيم النص إلى "قطع فرعية" (Child Chunks) أصغر (مثلاً، 512 توكن) مناسبة لعملية التضمين (Embedding).
3. **الخطوة 3: الاسترجاع الهجين:** يتم تضمين (Embed) وفهرسة *القطع الفرعية* فقط (لضمان دقة البحث الدلالي). عند الاستعلام، يتم استرجاع القطع الفرعية الأكثر صلة، ولكن يتم تمرير *قطع الأصل* (Parent Chunks) الكاملة المرتبطة بها إلى سياق النموذج اللغوي.

هذا النهج يمنحك دقة البحث (من الأبناء) مع الحفاظ على السياق الكامل (من الآباء).

**جدول 3: مصفوفة استراتيجية التقطيع (Chunking Strategy Matrix) للمستندات الأكاديمية**

| **الاستراتيجية** | **نوع المستند المناسب** | **دقة الاسترجاع** | **الوعي بالسياق** | **تعقيد التنفيذ** |
| --- | --- | --- | --- | --- |
| **ثابت الحجم (Fixed-Size)** 50 | نص غير منظم | منخفض | منخفض (يفصل السياق) | منخفض جداً |
| **متكرر (Recursive)** 50 | نص برمجي، Markdown | متوسط | متوسط | منخفض |
| **دلالي (Semantic)** 46 | نثر كثيف (أوراق بحثية) | عالي | عالي (لكنه قد يدمج أقساماً) | متوسط |
| **هرمي (Parent-Child)** 47 | مستندات منظمة (سيلابوس، لوائح) 44 | عالي جداً | عالي جداً (يحافظ على الهيكل) | عالي |

### 3.3. ضمان الدقة وتقليل الهلوسة (Hallucination)

الخطر الأكبر في الإرشاد الأكاديمي هو "الهلوسة" (Hallucination).51 إعطاء الطالب معلومة خاطئة عن متطلبات التخرج أو قواعد التسجيل هو أمر غير مقبول.

على الرغم من أن RAG مصمم لـ "تثبيت" (Ground) النموذج اللغوي في "بيانات واقعية" 51، إلا أن هذا لا يضمن عدم الهلوسة بنسبة 100%.

لتحقيق الدقة المطلوبة، يجب تنفيذ نمط هجين يُعرف بـ "URAG" (Unified RAG).55 هذا النظام يدمج "أنظمة قائمة على القواعد (Rule-Based)" مع "RAG" في مسار ذي مستويين للأسئلة الواقعية (Factual Questions).55

آلية العمل كالتالي:

1. يطرح الطالب سؤالاً حرجاً: "ما هي الدرجة المطلوبة لاجتياز مادة الفيزياء 101؟".55
2. **المستوى 1: محرك القواعد (Rule-Based Engine / FAQ):** قبل استدعاء أي نموذج لغوي، يتم تمرير السؤال أولاً إلى "نظام أسئلة وأجوبة متكررة (FAQ) شامل" ومنسق يدوياً.55 هذا النظام يحتوي على "إجابات ذهبية" للحقائق الثابتة (مثل قواعد النجاح، الرسوم الدراسية، المواعيد النهائية).
3. **القرار:** إذا تم العثور على تطابق عالي الثقة (e.g., تطابق نمط أو regex لـ "درجة النجاح" + "PHYS101")، يتم إرجاع الإجابة المحددة مسبقاً *فوراً* ("درجة C أو أعلى"). **لا يتم استدعاء النموذج اللغوي على الإطلاق.**
4. **المستوى 2: RAG:** "إذا لم يتم العثور على تطابق في الأسئلة الشائعة" 55 (مثل سؤال مفتوح: "اشرح لي نظرية النسبية")، عندها فقط يتم تمرير السؤال إلى مسار RAG العادي (الذي تم وصفه في 3.2) لاسترجاع المستندات وتوليد إجابة ديناميكية.

هذا النهج الهجين 56 يمنحك *أمان* و *دقة* النظام القائم على القواعد للحقائق الثابتة، و *مرونة* RAG للأسئلة المفتوحة.

## القسم 4: تنفيذ الميزات السوبر المميزة: النمذجة التنبؤية (Predictive Modeling) لنجاح الطالب

تتطلب الميزات "السوبر" بناء نماذج تعلم آلة كلاسيكية للتنبؤ بنجاح الطالب وتحديد الطلاب المعرضين للخطر.

### 4.1. المكدس التقني والمكتبات الأساسية

المكدس التقني لهذه المهمة هو معيار صناعي في علوم البيانات:

* **Python** 58 كلغة أساسية.
* **Pandas** 59 لمعالجة البيانات وهندسة الميزات.
* **Scikit-learn (sklearn)** 4 لبناء وتقييم النماذج التنبؤية.

تشير الأبحاث في مجال التنبؤ الأكاديمي إلى أن نماذج مثل **Random Forest** (الغابة العشوائية) 62 و **XGBoost** 58 تُظهر فعالية عالية في تصنيف الطلاب (e.g., ناجح، متعثر، متسرب).65

### 4.2. هندسة الميزات (Feature Engineering): المصدر الحقيقي للقوة التنبؤية

نجاح هذه الميزة "السوبر" لا يعتمد على اختيار الخوارزمية (Random Forest كافٍ 62)، بل يعتمد بالكامل على جودة *هندسة الميزات* (Feature Engineering).63 يجب أن يركز العمل على جمع ودمج مصادر بيانات متنوعة 66:

1. **بيانات التسجيل (Enrollment Data):** "المسار الأكاديمي، الديموغرافيا، والعوامل الاجتماعية والاقتصادية".65
2. **البيانات السلوكية (LMS Data):** "بيانات تسجيل الدخول ودرجات نظام إدارة التعلم (LMS)".67
3. **البيانات الأكاديمية (Academic Data):** "الأداء السابق" 68 والدرجات في المواد الإلزامية والمتطلبات المسبقة.69

النماذج البسيطة التي تعتمد فقط على "الأداء السابق" قد تكون دقيقة في التنبؤ بالنجاح/الفشل، لكنها "لا توفر معلومات دقيقة" لتحديد *درجة* الخطر أو سببه.68 النموذج القوي هو الذي يمكنه استيعاب 50+ متغيراً 70 لإنشاء ميزات مركبة (مثل: معدل\_تسجيل\_الدخول\_الأسبوعي، الانحراف\_عن\_متوسط\_درجات\_المتطلب\_المسبق).

### 4.3. التطبيق العملي: من التنبؤ إلى التدخل (دراسة حالة GSU)

من الأهمية بمكان فهم *كيفية* استخدام هذا النموذج التنبؤي. الاستخدام الخاطئ هو أن يقوم الذكاء الاصطناعي بإبلاغ الطالب مباشرة: "أنت معرض لخطر الرسوب". هذا غير فعال وقد يكون غير أخلاقي.71

التطبيق الصحيح والفعال هو نمط "المرشد في الحلقة" (Advisor-in-the-Loop)، وهو ما طبقته بنجاح جامعة ولاية جورجيا (GSU).69

1. **التتبع (Tracking):** يستخدم نظام GSU التحليلات التنبؤية لتتبع "800 عامل خطر" لأكثر من 40,000 طالب يومياً.69
2. **التنبيه (Alerting):** النظام لا يتنبأ فقط، بل "يولد تنبيهات (Alerts)" للمرشدين الأكاديميين.69
   * *مثال 1:* "الطالب 'س' سجل في مساق لا ينطبق على شهادته".69
   * *مثال 2:* "الطالب 'ص' حصل على 'C' في مادة الرياضيات... وهذا ليس جيداً بما يكفي للنجاح في مادة الكيمياء العليا التي سجل فيها".69
3. **التدخل (Intervention):** بناءً على التنبيه، "يصل مرشد (بشري) إلى الطالب في اليوم التالي".69 الهدف هو تحويل الإرشاد من كونه "رد فعل" (Reactive) إلى "استباقي" (Proactive) 72، والتدخل لمساعدة الطالب "عند أول علامة على وجود مشكلة" 69، وليس بعد فشله.

لذلك، يجب أن تكون مخرجات نموذجك التنبؤي (القسم 4) عبارة عن "لوحة تحكم للمرشدين" (Advisor Dashboard)، وليس رسالة مباشرة للطالب.

### 4.4. الاعتبارات الأخلاقية: الشفافية وتجنب التحيز

هناك خطر حقيقي من أن هذه النماذج التنبؤية قد "تعزز عن غير قصد عدم المساواة التعليمية" 75 أو "تقنن التحيز (codify bias)" 76 الموجود في البيانات التاريخية (مثل التحيزات ضد مجموعات ديموغرافية معينة).

الحل يكمن في "الشفافية (Transparency)" و "القابلية للتفسير (Interpretability)".66 يجب ألا يكون النموذج "صندوقاً أسود" (Black Box) 71 بالنسبة للمرشد:

* عندما يُصدر النظام تنبيهاً لطالب معين، يجب أن يعرض *لماذا*.
* النماذج المستخدمة (مثل Random Forest و XGBoost) 63 لها ميزة قوية: يمكنها حساب "أهمية الميزات (Feature Importance)".63
* يجب أن تعرض لوحة التحكم للمرشد: "تم التنبيه بشأن الطالب 'س' بسبب: 1. انخفاض درجة المتطلب المسبق (40%)، 2. انخفاض معدل تسجيل الدخول إلى LMS (30%)، 3....".

هذا يحقق "محو الأمية البياناتية" (Data Literacy) للمرشد 70، مما يسمح له بتطبيق حكمه البشري، والتحقق من التحيز المحتمل، وتحديد أفضل مسار للتدخل بناءً على رؤية كاملة.

## القسم 5: تنفيذ الميزات فائقة الذكاء: دمج الرسوم البيانية (Graph RAG) ومعالجة اللغات الطبيعية المتقدمة (NLP)

تمثل ميزاتك "فائقة الذكاء" (مثل "شجرة المهارات" وفهم مسارات المتطلبات المسبقة) تحدياً يتجاوز قدرات RAG التقليدي.

### 5.1. لماذا نحتاج إلى Graph RAG؟ حدود RAG التقليدي

RAG التقليدي (القسم 3) يعتمد على البحث الدلالي (Semantic Vector Search). إنه يجيب على أسئلة "التشابه" (e.g., "ما هي المستندات التي تتحدث عن الذكاء الاصطناعي؟").

ومع ذلك، RAG التقليدي "يكافح لالتقاط... العلاقات بين الكيانات".77 أسئلة مثل: "ما هي المواد التي يجب أن آخذها *بعد* 'هياكل البيانات' للوصول إلى تخصص 'الذكاء الاصطناعي'؟" هي *مشاكل علاقات* (Relationship Problems)، وليست مشاكل تشابه دلالي.

هنا يأتي دور قواعد بيانات الرسم البياني (Graph Databases) مثل **Neo4j** 77، المصممة خصيصاً لـ "تمثيل وتخزين معلومات متنوعة ومترابطة" 77 (مثل: (مادة: هياكل بيانات) --> (مادة: خوارزميات)).

النهج الأقوى، المعروف باسم "GraphRAG" 78 أو "HybridRAG" 81، يدمج قوة البحث الدلالي (Vector DBs) مع قوة "الاستدلال متعدد الخطوات (Multi-hop Reasoning)" (Graph DBs).80 يمكن لهذا النظام تتبع المسارات والعلاقات في قاعدة المعرفة للإجابة على الأسئلة المعقدة التي يفشل فيها RAG التقليدي.83

### 5.2. بناء Graph RAG: Neo4j و LangChain

المكدس التقني المثالي لتنفيذ هذا هو **Neo4j** كقاعدة بيانات رسم بياني و **LangChain** كمنسق.47

لدى LangChain مكونات مدمجة مثل GraphCypherQAChain.84 تعمل هذه السلسلة عن طريق أخذ سؤال المستخدم باللغة الطبيعية واستخدام نموذج لغوي (LLM) *لتوليد استعلام Cypher* (لغة استعلام Neo4j).84 يتم إرسال هذا الاستعلام إلى Neo4j لجلب البيانات العلائقية الدقيقة، ثم يتم تمرير النتائج مرة أخرى إلى النموذج اللغوي لصياغة إجابة باللغة الطبيعية.

للتطبيقات الأكثر تقدماً، يمكن استخدام **LangGraph** (امتداد لـ LangChain) 84 لإنشاء "عملاء أذكياء" (Agents). يمكن لهذه العملاء تنفيذ "تجزئة" (Decomposition) للسؤال 84 وتوجيهه (Routing) بذكاء.

على سبيل المثال، سؤال مركب مثل: "ماذا قال الأستاذ 'أحمد' عن 'الشبكات العصبونية' في مادة 'الذكاء الاصطناعي'، وما هي المتطلبات المسبقة لهذه المادة؟"

سيقوم "الموجه" (Router) 84 بتقسيم هذا إلى:

1. **سؤال فرعي 1 (دلالي):** "ماذا قال الأستاذ 'أحمد' عن 'الشبكات العصبونية'؟" -> **التوجيه إلى:** ChromaDB (للبحث الدلالي في نصوص المحاضرات).86
2. **سؤال فرعي 2 (رسم بياني):** "ما هي المتطلبات المسبقة لمادة 'الذكاء الاصطناعي'؟" -> **التوجيه إلى:** Neo4j (للبحث عن علاقة IS\_PREREQ\_FOR).86

يقوم LangGraph بعد ذلك بتجميع الإجابات من كلا المصدرين 82 وتقديمها كـ "سياق" موحد للنموذج اللغوي لإنشاء إجابة واحدة شاملة.

### 5.3. خط أنابيب معالجة اللغات الطبيعية المتقدم (Advanced NLP Pipeline): استخراج المهارات وتحليل المشاعر

لتحقيق "الذكاء الفائق"، يحتاج النظام إلى خطي أنابيب لمعالجة اللغات الطبيعية (NLP): واحد لـ "وقت الاستيعاب" (Ingestion-Time) وواحد لـ "وقت التشغيل" (Runtime).

1. خط أنابيب الاستيعاب (Ingestion-Time Pipeline): بناء الرسم البياني تلقائياً  
   كيف يتم ملء قاعدة بيانات Neo4j؟ يدوياً؟ لا.  
   يتم استخدام مكتبة spaCy، وهي "المعيار الصناعي" لـ "استخراج المعلومات" (Information Extraction).5 (يُفضل spaCy على NLTK 88 لأنه أسرع ومصمم للإنتاج 89).  
   عندما يتم تحميل "سيلابوس" (Syllabus) جديد إلى النظام، يتم تشغيل خط أنابيب spaCy NER (التعرف على الكيانات المسماة) 91 عليه. يتم تدريب هذا الخط على استخراج الكيانات الأكاديمية: (SKILL: 'PyTorch')، (CONCEPT: 'Regression')، (PREREQ: 'CS101').  
   تُستخدم هذه الكيانات المستخرجة 92 لإنشاء العقد (Nodes) والعلاقات (Edges) تلقائياً في قاعدة بيانات Neo4j.79 هذا يحل مشكلة بناء قاعدة المعرفة البيانية بكفاءة.
2. خط أنابيب وقت التشغيل (Runtime Pipeline): تحليل المشاعر  
   لجعل النموذج التنبؤي (القسم 4) أكثر ذكاءً، يمكننا تحليل مشاعر الطلاب.93  
   يتم استخدام مكتبة VADER.94 VADER "محسَّن لبيانات وسائل التواصل الاجتماعي" (والنصوص غير الرسمية مثل منتديات الطلاب) 94 ويتفوق عموماً على TextBlob في هذه السيناريوهات.96  
   يقوم النظام بمراقبة منتديات نقاش الطلاب. يتم تشغيل VADER 94 على جميع المنشورات المتعلقة بمادة "CS101" لحساب درجة مشاعر مجمعة.  
   الربط: يتم إرسال نتيجة المشاعر المجمعة (e.g., CS101\_Sentiment: -0.4) مرة أخرى إلى نموذج التنبؤ (القسم 4) كـ ميزة جديدة (new feature). هذا يربط النظام "فائق الذكاء" بالنظام "السوبر"، مما يجعل التنبؤ بالخطر أكثر دقة (e.g., "الطالب معرض للخطر + مشاعر المادة سلبية جداً" -> تنبيه عالي الأولوية).

## القسم 6: خطة النشر والتشغيل الكاملة (Deployment Blueprint)

يربط هذا القسم جميع المفاهيم السابقة في هيكل ملفات ودليل تكوين عملي.

### 6.1. هيكل المشروع المقترح (Project Structure)

بناءً على الممارسات الفضلى لتطبيقات RAG متعددة الحاويات 98، يُقترح الهيكل التالي:

/my-ai-advisor  
├── docker-compose.yml <-- ملف التنسيق الرئيسي   
├──.env <-- لتخزين المفاتيح والأسرار   
├── /frontend <-- خدمة Streamlit   
│ ├── Dockerfile.frontend  
│ ├── app.py <-- كود واجهة المستخدم  
│ └── requirements.txt <-- (streamlit, requests)  
├── /backend <-- خدمة FastAPI   
│ ├── Dockerfile.backend  
│ ├── main.py <-- كود الواجهة الخلفية (API Endpoints)  
│ └── requirements.txt <-- (fastapi, uvicorn, langchain,...)  
├── /data <-- المجلد الذي يحتوي على ملفات PDF/المعرفة  
├── /vectordb\_storage <-- مجلد التخزين الدائم لـ ChromaDB   
└── /graphdb\_storage <-- مجلد التخزين الدائم لـ Neo4j

### 6.2. ملف Dockerfile: الواجهة الأمامية (Streamlit)

بناءً على معماريتنا المنفصلة (الرؤية 1.2)، يجب أن تكون حاوية الواجهة الأمامية "خفيفة" (Lean). العديد من الأمثلة 106 تضع مكتبات ثقيلة مثل torch في حاوية Streamlit. يجب *تجنب* هذا.

يجب أن يكون ملف requirements.txt الخاص بالواجهة الأمامية بسيطاً جداً: streamlit و requests.11

**frontend/Dockerfile.frontend (مثال):**

Dockerfile

# استخدم صورة بايثون خفيفة  
FROM python:3.9-slim  
  
# تعيين دليل العمل  
WORKDIR /app  
  
# نسخ ملف المتطلبات أولاً للاستفادة من التخزين المؤقت لـ Docker  
COPY requirements.txt.  
RUN pip install --no-cache-dir -r requirements.txt  
  
# نسخ باقي كود التطبيق  
COPY..  
  
# عرض المنفذ الافتراضي لـ Streamlit  
EXPOSE 8501  
  
# الأمر الافتراضي لتشغيل التطبيق  
CMD ["streamlit", "run", "app.py", "--server.port=8501", "--server.address=0.0.0.0"]

(مستوحى من 6)

### 6.3. ملف Dockerfile: الواجهة الخلفية (FastAPI)

ستكون هذه هي الحاوية "الثقيلة" التي تحتوي على جميع مكتبات الذكاء الاصطناعي. سيتبع ملف Dockerfile.backend هيكلاً مشابهاً ( FROM python:3.9، WORKDIR /app، COPY requirements.txt.، RUN pip install...).101

لكن ملف requirements.txt الخاص به سيحتوي على كل شيء آخر:

fastapi, uvicorn, langchain, langchain-community, langchain-neo4j, neo4j, chromadb-client, scikit-learn, pandas, spacy, vaderSentiment, torch, transformers (لنماذج التضمين)، ollama.

قد يتطلب هذا الملف أيضاً تثبيت تبعيات النظام، مثل RUN apt-get update && apt-get install -y build-essential curl git.6

### 6.4. ملف docker-compose.yml: المخطط الرئيسي للتنسيق (The Master Blueprint)

هذا هو الملف المركزي الذي يربط جميع الخدمات الخمس.30 نجاح تشغيل هذا النظام يعتمد على ثلاثة تكوينات حرجة: الشبكات، الوصول إلى GPU، واستمرارية البيانات.

**الشبكات (Networking):** الخطأ الأكثر شيوعاً هو فشل اتصال الحاويات ببعضها البعض.99 عندما تحاول حاوية Streamlit الاتصال بـ http://localhost:8000، فإنها تتصل بـ *نفسها*، وليس بحاوية FastAPI. الحل هو استخدام أسماء الخدمات التي يحددها Docker Compose. يجب تمرير متغير بيئة (Environment Variable) إلى خدمة frontend 98 مثل FASTAPI\_BACKEND\_URL=http://backend:8000.

**الوصول إلى GPU (GPU Access):** افتراضياً، لا يمكن لحاويات Docker "رؤية" وحدة معالجة الرسومات (GPU). إذا لم يتم تكوين هذا، سيعمل نموذج Ollama على وحدة المعالجة المركزية (CPU) ويكون بطيئاً جداً. الحل هو إضافة قسم deploy لخدمة ollama لتمكين nvidia.29

**استمرارية البيانات (Persistence):** إذا تم إيقاف الحاويات وإعادة تشغيلها، ستضيع جميع البيانات (فهرس المتجهات، الرسم البياني للمعرفة). الحل هو استخدام volumes 101 لربط مجلدات داخل الحاوية (e.g., /chroma) بمجلدات على الجهاز المضيف (e.g., ./vectordb\_storage).101

**جدول 4: تعريفات خدمة docker-compose.yml المقترحة**

| **الخدمة (Service)** | **build / image** | **ports** | **volumes (للاستمرارية)** | **environment (متغيرات هامة)** | **deploy (للوصول إلى GPU)** |
| --- | --- | --- | --- | --- | --- |
| **frontend** | build:./frontend 98 | "8501:8501" | - | FASTAPI\_BACKEND\_URL=http://backend:8000 98 | - |
| **backend** | build:./backend 98 | "8000:8000" | ./data:/app/data (للوصول لملفات PDF) | OLLAMA\_BASE\_URL=http://llm-service:11434 98  CHROMA\_HOST=vector-db 101  NEO4J\_URI=bolt://graph-db:7687 | - |
| **llm-service** | image: ollama/ollama 29 | "11434:11434" | ollama\_data:/root/.ollama | - | resources: {reservations: {devices: [{driver: nvidia, count: 1, capabilities: [gpu]}]}} 29 |
| **vector-db** | image: chromadb/chroma 101 | "8001:8000" | chroma\_data:/chroma 101 (للتخزين الدائم) | ANONYMIZED\_TELEMETRY=False | - |
| **graph-db** | image: neo4j:latest | "7687:7687"  "7474:7474" | neo4j\_data:/data | NEO4J\_AUTH=neo4j/password (مثال) | - |

*ملاحظة: يجب تعريف volumes: {ollama\_data: {}, chroma\_data: {}, neo4j\_data: {}} في المستوى الجذر للملف لإدارة التخزين الدائم.*

## الخاتمة: خارطة الطريق للتشغيل الفعّال

يقدم هذا التقرير مخططاً معمارياً شاملاً وقابلاً للتنفيذ لمشروع "مرشدي الأكاديمي الذكي". من خلال اعتماد مكدس تقني حديث ومُحَوّى (Containerized) يتكون من **Streamlit** (للواجهة)، **FastAPI** (للمنطق)، **Ollama** (للنماذج اللغوية)، **ChromaDB** (للبحث الدلالي)، و **Neo4j** (للبحث في العلاقات)، يمكن بناء نظام قوي وقابل للتطوير.

الأساس في هذا التصميم هو ملف docker-compose.yml 100، الذي يعمل كـ "المخطط الرئيسي" الذي ينسق جميع هذه الخدمات.30 تضمن هذه المعمارية المنفصلة (Decoupled) 9 أن يكون النظام فعالاً وسهل الصيانة.

يسمح التصميم المعياري (Modular) بتنفيذ الميزات على مراحل:

1. **البدء بالميزات الأساسية (RAG)** عن طريق ربط FastAPI بـ Ollama و ChromaDB.30
2. **إضافة الميزات السوبر (Predictive)** عن طريق إضافة نماذج Scikit-learn إلى خدمة FastAPI واستخدامها لتشغيل لوحة تحكم للمرشدين.105
3. **توسيع إلى الميزات فائقة الذكاء (Graph RAG)** عن طريق دمج خدمة Neo4j واستخدام LangChain لتوجيه الاستعلامات المعقدة.78

التوصية النهائية هي البدء ببناء ملف docker-compose.yml (كما هو مفصل في الجدول 4) كنقطة انطلاق، حيث يمثل حجر الأساس الذي يربط جميع المكونات معاً في نظام واحد متكامل وفعال.

#### المصادر التي تم الاقتباس منها

1. What's the difference between Docker and Python virtualenv? - Stack Overflow, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://stackoverflow.com/questions/50974960/whats-the-difference-between-docker-and-python-virtualenv>
2. Docker مقابل VM - الفرق بين تقنيات نشر التطبيقات - Amazon AWS, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://aws.amazon.com/ar/compare/the-difference-between-docker-vm/>
3. what's the difference between a python virtual environment vs a docker container? - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/docker/comments/11jiamo/whats_the_difference_between_a_python_virtual/>
4. Machine learning in Python with scikit-learn - Fun MOOC, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.fun-mooc.fr/en/courses/machine-learning-python-scikit-learn/>
5. spaCy · Industrial-strength Natural Language Processing in Python, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://spacy.io/>
6. Deploy Streamlit using Docker, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.streamlit.io/deploy/tutorials/docker>
7. What is Docker?, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.docker.com/get-started/docker-overview/>
8. Build and Deploy a LangChain-Powered Chat App with Docker and Streamlit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.docker.com/blog/build-and-deploy-a-langchain-powered-chat-app-with-docker-and-streamlit/>
9. Machine Learning Model deployment with FastAPI, Streamlit and Docker - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/latinxinai/fastapi-and-streamlit-app-with-docker-compose-e4d18d78d61d>
10. Building a dashboard in Python using Streamlit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://blog.streamlit.io/crafting-a-dashboard-app-in-python-using-streamlit/>
11. Introduction to Building AI-Powered Apps with Streamlit and FastAPI - DEV Community, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://dev.to/cypriantinasheaarons/introduction-to-building-ai-powered-apps-with-streamlit-and-fastapi-73d>
12. Concurrency and async / await - FastAPI, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://fastapi.tiangolo.com/async/>
13. Streamlit as a Session State Machine: Enhancing Interactive Web Apps | by BigCodeGen, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://bigcodegen.medium.com/streamlit-as-a-session-state-machine-enhancing-interactive-web-apps-b831b3794df8>
14. From Backend To Frontend: Connecting FastAPI And Streamlit ..., تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://pybit.es/articles/from-backend-to-frontend-connecting-fastapi-and-streamlit/>
15. Building a Website with Python, FastAPI, and Streamlit | by Obafemi - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@obaff/building-a-website-with-python-fastapi-and-streamlit-418f48c41af2>
16. Issue with integrating fastapi with streamlit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://discuss.streamlit.io/t/issue-with-integrating-fastapi-with-streamlit/66888>
17. Build a basic LLM chat app - Streamlit Docs, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.streamlit.io/develop/tutorials/chat-and-llm-apps/build-conversational-apps>
18. Session State - Streamlit Docs, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.streamlit.io/develop/api-reference/caching-and-state/st.session_state>
19. Add statefulness to apps - Streamlit Docs, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.streamlit.io/develop/concepts/architecture/session-state>
20. How to create a chat history on the side bar just like chatGPT? - LLMs and AI - Streamlit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://discuss.streamlit.io/t/how-to-create-a-chat-history-on-the-side-bar-just-like-chatgpt/59492>
21. Ollama vs. vLLM: A deep dive into performance benchmarking | Red Hat Developer, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://developers.redhat.com/articles/2025/08/08/ollama-vs-vllm-deep-dive-performance-benchmarking>
22. Ollama vs. vLLM: The Definitive Guide to Local LLM Frameworks in 2025, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://blog.alphabravo.io/ollama-vs-vllm-the-definitive-guide-to-local-llm-frameworks-in-2025/>
23. 6 Powerhouse Tools for Running LLMs Locally: Your Guide to Offline AI Mastery - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/newaitools/6-powerhouse-tools-for-running-llms-locally-your-guide-to-offline-ai-mastery-9ea42a7f4942>
24. Performance vs Practicality: A Comparison of vLLM and Ollama | by Robert McDermott, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://robert-mcdermott.medium.com/performance-vs-practicality-a-comparison-of-vllm-and-ollama-104acad250fd>
25. Ollama vs vLLM: A Detailed Comparison of LLM Frameworks - DEV Community, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://dev.to/mechcloud_academy/ollama-vs-vllm-a-detailed-comparison-of-llm-frameworks-513m>
26. Performance wise what is the best backend right now? : r/LocalLLaMA - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/LocalLLaMA/comments/1nwpxxx/performance_wise_what_is_the_best_backend_right/>
27. Run LLMs Locally with Docker: A Quickstart Guide to Model Runner, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.docker.com/blog/run-llms-locally/>
28. Build Full Stack LLM Chat App with Docker Model Runner, LangChain and Streamlit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.youtube.com/watch?v=oIqF0z2UhDM>
29. Docker: Streamlit, Qdrant, Ollama — RAG | by Ion Stefanache - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@ion.stefanache0/docker-flask-rag-114c795de22c>
30. Dockerizing a RAG Application with FastAPI, LlamaIndex, Qdrant and Ollama, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://otmaneboughaba.com/posts/dockerize-rag-application/>
31. Which Quantization Method Is Best for You?: GGUF, GPTQ, or AWQ | E2E Networks Blog, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.e2enetworks.com/blog/which-quantization-method-is-best-for-you-gguf-gptq-or-awq>
32. LLMs on CPU: The Power of Quantization with GGUF, AWQ, & GPTQ - Ionio, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.ionio.ai/blog/llms-on-cpu-the-power-of-quantization-with-gguf-awq-gptq>
33. A Comprehensive Analysis of Post-Training Quantization Strategies for Large Language Models: GPTQ, AWQ, and GGUF | Uplatz Blog, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://uplatz.com/blog/a-comprehensive-analysis-of-post-training-quantization-strategies-for-large-language-models-gptq-awq-and-gguf/>
34. GPU Requirement Guide for Llama 3 (All Variants) - ApX Machine Learning, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://apxml.com/posts/ultimate-system-requirements-llama-3-models>
35. LLaMA 7B GPU Memory Requirement - Transformers - Hugging Face Forums, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://discuss.huggingface.co/t/llama-7b-gpu-memory-requirement/34323>
36. What should be the minimum spec to run Llama 3.1 8B in a local server? - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/LocalLLaMA/comments/1eavuzr/what_should_be_the_minimum_spec_to_run_llama_31/>
37. Hardware specs for GGUF 7B/13B/30B parameter models · ggml-org ..., تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://github.com/ggml-org/llama.cpp/discussions/3847>
38. (PDF) Intelligent Q&A application of rules and regulations based on large models and RAG technology - ResearchGate, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.researchgate.net/publication/395769554_Intelligent_QA_application_of_rules_and_regulations_based_on_large_models_and_RAG_technology>
39. The RAG Showdown: LangChain vs. LlamaIndex — Which Tool Reigns Supreme? | by Ajay Verma | Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@ajayverma23/the-rag-showdown-langchain-vs-llamaindex-which-tool-reigns-supreme-f79f6fe80f86>
40. 5 Python Libraries to Build an Optimized RAG System - MachineLearningMastery.com, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://machinelearningmastery.com/5-python-libraries-build-optimized-rag-system/>
41. Start today, which Python packages would you use for RAG - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/Rag/comments/1f9m19m/start_today_which_python_packages_would_you_use/>
42. Build a RAG agent with LangChain, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.langchain.com/oss/python/langchain/rag>
43. 11 Chunking Strategies for RAG — Simplified & Visualized | by Mastering LLM (Large Language Model), تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://masteringllm.medium.com/11-chunking-strategies-for-rag-simplified-visualized-df0dbec8e373>
44. Advanced Chunking/Retrieving Strategies for Legal Documents : r/Rag - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/Rag/comments/1jdi4sg/advanced_chunkingretrieving_strategies_for_legal/>
45. Best Chunking Strategy for the Medical RAG System (Guidelines Docs) in PDFs - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/Rag/comments/1ljhksy/best_chunking_strategy_for_the_medical_rag_system/>
46. Chunking Strategies to Improve Your RAG Performance | Weaviate, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://weaviate.io/blog/chunking-strategies-for-rag>
47. LangChain Neo4j Integration - Neo4j Labs, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://neo4j.com/labs/genai-ecosystem/langchain/>
48. Building a Document Q&A System with FastAPI, ChromaDB & LLaMA 3 on GPU - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@purnageria8/building-a-document-q-a-system-with-fastapi-chromadb-llama-3-on-gpu-c725b2d9ed3b>
49. Develop a RAG Solution - Chunking Phase - Azure Architecture Center | Microsoft Learn, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/rag/rag-chunking-phase>
50. Implement RAG chunking strategies with LangChain and watsonx.ai - IBM, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.ibm.com/think/tutorials/chunking-strategies-for-rag-with-langchain-watsonx-ai>
51. RAG Hallucination: What is It and How to Avoid It, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.k2view.com/blog/rag-hallucination/>
52. Hallucination‐Free? Assessing the Reliability of Leading AI Legal Research Tools - Daniel E. Ho - Stanford University, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://dho.stanford.edu/wp-content/uploads/Legal_RAG_Hallucinations.pdf>
53. Developing Retrieval Augmented Generation (RAG) based LLM Systems from PDFs: An Experience Report - arXiv, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://arxiv.org/html/2410.15944v1>
54. Reducing hallucinations in large language models with custom intervention using Amazon Bedrock Agents | Artificial Intelligence, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://aws.amazon.com/blogs/machine-learning/reducing-hallucinations-in-large-language-models-with-custom-intervention-using-amazon-bedrock-agents/>
55. URAG: Implementing a Unified Hybrid RAG for Precise Answers in University Admission Chatbots – A Case Study at HCMUT - arXiv, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://arxiv.org/html/2501.16276v1>
56. Hybrid RAG: Boosting RAG Accuracy - Research AIMultiple, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://research.aimultiple.com/hybrid-rag/>
57. Open-Sourcing Deterministic Rule-based Retrieval for RAG | by Chia Jeng Yang - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/enterprise-rag/open-sourcing-rule-based-retrieval-677946260973>
58. Predicting Student Success with Machine Learning: A Complete Guide to Implementation | by Ranga Hande | Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@ranga4all1_/predicting-student-success-with-machine-learning-a-complete-guide-to-implementation-b199f76e8d51>
59. Predicting Student Success Rate ( Acc = 0.92% ) - Kaggle, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.kaggle.com/code/ranzeet013/predicting-student-success-rate-acc-0-92>
60. Unveiling the Secret to Academic Success: Predicting Student Performance with ML Linear Regression | by Vipin Singh Inkiya | Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@vipinnation/unveiling-the-secret-to-academic-success-predicting-student-performance-with-ml-linear-regression-88f4ea73f36f>
61. scikit-learn: machine learning in Python — scikit-learn 1.7.2 documentation, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://scikit-learn.org/>
62. Predictive Modeling for Student Performance - Kaggle, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.kaggle.com/code/nileshely/predictive-modeling-for-student-performance>
63. Study on Feature Engineering and Ensemble Learning for Student Academic Performance Prediction - The Science and Information (SAI) Organization, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://thesai.org/Publications/ViewPaper?Volume=13&Issue=5&Code=IJACSA&SerialNo=58>
64. Predictive Analysis of Students' Learning Performance Using Data Mining Techniques: A Comparative Study of Feature Selection Methods - MDPI, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.mdpi.com/2571-5577/6/5/86>
65. Predict Students' Dropout and Academic Success - UCI Machine Learning Repository, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success>
66. Predictive Models for Educational Purposes: A Systematic Review - MDPI, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.mdpi.com/2504-2289/8/12/187>
67. Feature Engineering on LMS Data to Optimize Student Performance Prediction - arXiv, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://arxiv.org/abs/2504.02916>
68. Identifying Students At Risk Using Prior Performance Versus a Machine Learning Algorithm, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://ies.ed.gov/use-work/resource-library/report/descriptive-study/identifying-students-risk-using-prior-performance-versus-machine-learning-algorithm>
69. Approaching Student Success With Predictive Analytics - Georgia State University, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://success.gsu.edu/approach/>
70. View of Implementing Predictive Analytics in Academic Advising - Journals - Penn State, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://journals.psu.edu/mentor/article/view/63332/62243>
71. Questioning Predictive Analytics for Academic Advising - NACADA - Kansas State University, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://nacada.ksu.edu/Resources/Academic-Advising-Today/View-Articles/Questioning-Predictive-Analytics-for-Academic-Advising.aspx>
72. Predicting Student Outcomes to Drive Proactive Support: An Exploration of Machine Learning to Advance Student Equity & Succe - Office of the Provost - University of Oregon, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://provost.uoregon.edu/sites/default/files/2021-12/final-brief-early-pred-retention.pdf>
73. V5I4: Predicting Risk Earlier: Using Artificial Intelligence to Support Student Success and Combat Inequity - NWCCU, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://nwccu.org/news/v5i4-predicting-risk-earlier-using-artificial-intelligence-to-support-student-success-and-combat-inequity/>
74. Using Predictive Analytics for Student Success and Retention at Community Colleges, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.liaisonedu.com/resources/blog/using-predictive-analytics-for-student-success-and-retention-at-community-colleges/>
75. Using AI to predict student success in higher education - Brookings Institution, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.brookings.edu/articles/using-ai-to-predict-student-success-in-higher-education/>
76. IV-A.3(3) The Promise and Peril of Predictive Analytics in Higher Education - Report Center, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://reportcenter.highered.texas.gov/agency-publication/miscellaneous/iv-a-3-3-the-promise-and-peril-of-predictive-analytics-in-higher-education/>
77. RAG with LangChain Part3: Graph RAG | by Mustafa604 | Nov, 2025, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@Mustafa77/rag-with-langchain-part3-graph-rag-239048b1beea>
78. Using a Knowledge Graph to implement a RAG application - Neo4j, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://neo4j.com/blog/developer/rag-tutorial/>
79. From Legal Documents to Knowledge Graphs | by Tomaz Bratanic | Neo4j Developer Blog, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/neo4j/from-legal-documents-to-knowledge-graphs-ccd9cb062320>
80. AI and Graph Databases: Enhancing Data Retrieval - Analytics Vidhya, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.analyticsvidhya.com/blog/2024/02/ai-and-graph-databases-enhancing-data-retrieval/>
81. HybridRAG and Why Combine Vector Embeddings with Knowledge Graphs for RAG?, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://memgraph.com/blog/why-hybridrag>
82. Leveraging Complex RAG Approaches: Combining Graph DB and Vector DB for Real-Life Applications | by Alimahroof | Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@alimahroof09/leveraging-complex-rag-approaches-combining-graph-db-and-vector-db-for-real-life-applications-9f19d9135846>
83. Graph RAG: Navigating graphs for Retrieval-Augmented Generation using Elasticsearch, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.elastic.co/search-labs/blog/rag-graph-traversal>
84. Create a Neo4j GraphRAG Workflow Using LangChain and LangGraph, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://neo4j.com/blog/developer/neo4j-graphrag-workflow-langchain-langgraph/>
85. Building an Agentic Graph RAG System with Neo4j: A Journey in Progress | by ADITHYA GIRIDHARAN | Sep, 2025, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@AdithyaGiridharan/building-an-agentic-graph-rag-system-with-neo4j-a-journey-in-progress-bbb69743e1af>
86. Chroma vs Neo4j on Vector Search Capabilities - Zilliz blog, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://zilliz.com/blog/chroma-vs-neo4j-a-comprehensive-vector-database-comparison>
87. spaCy 101: Everything you need to know · spaCy Usage Documentation, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://spacy.io/usage/spacy-101>
88. Natural Language Processing With Python's NLTK Package - Real Python, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://realpython.com/nltk-nlp-python/>
89. Natural Language Processing With spaCy in Python - Real Python, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://realpython.com/natural-language-processing-spacy-python/>
90. NLTK vs spaCy: A Deeper Dive into NLP Libraries | by Jay | Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@seaflux/nltk-vs-spacy-a-deeper-dive-into-nlp-libraries-1416dbc5ac7f>
91. Natural Language Processing (NLP) Tutorial - GeeksforGeeks, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.geeksforgeeks.org/nlp/natural-language-processing-nlp-tutorial/>
92. Matching Skills and Candidates with Graph RAG - Ontotext, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.ontotext.com/blog/matching-skills-and-candidates-with-graph-rag/>
93. Sentiment Analysis Based on Online Course Feedback Using Textblob and Machine Learning Techniques - ResearchGate, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.researchgate.net/publication/378543286_Sentiment_Analysis_Based_on_Online_Course_Feedback_Using_Textblob_and_Machine_Learning_Techniques>
94. Sentiment Analysis in Python: TextBlob vs Vader Sentiment vs Flair ..., تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair>
95. Sentiment Analysis in Python: Textblob vs Vader? - Stack Overflow, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://stackoverflow.com/questions/47760662/sentiment-analysis-in-python-textblob-vs-vader>
96. Sentiment Analysis of Students' Feedback with NLP and Deep Learning: A Systematic Mapping Study - MDPI, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.mdpi.com/2076-3417/11/9/3986>
97. Pre-trained Sentiment Analysis Libraries for Python - Textblob or NLTK? - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/LanguageTechnology/comments/aabfbq/pretrained_sentiment_analysis_libraries_for/>
98. diicellman/dspy-rag-fastapi: FastAPI wrapper around DSPy - GitHub, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://github.com/diicellman/dspy-rag-fastapi>
99. ConnectionError on multi-Docker app (Streamlit + fastAPI) when deployed on Heroku, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://discuss.streamlit.io/t/connectionerror-on-multi-docker-app-streamlit-fastapi-when-deployed-on-heroku/42382>
100. Containerize a RAG application - Docker Docs, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.docker.com/guides/rag-ollama/containerize/>
101. Run Chroma DB on a local machine and as a Docker container. | by Abhishek V Tatachar, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://abhishektatachar.medium.com/run-chroma-db-on-a-local-machine-and-as-a-docker-container-a9d4b91d2a97>
102. How to create a Dockerfile for streamlit app - Stack Overflow, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://stackoverflow.com/questions/73063486/how-to-create-a-dockerfile-for-streamlit-app>
103. Use containers for RAG development - Docker Docs, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.docker.com/guides/rag-ollama/develop/>
104. Chroma | LlamaIndex Python Documentation, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://developers.llamaindex.ai/python/examples/vector_stores/chromaindexdemo/>
105. Using machine learning to improve student success in higher education - McKinsey, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.mckinsey.com/industries/education/our-insights/using-machine-learning-to-improve-student-success-in-higher-education>
106. Deploy Streamlit App to Hugging Face Spaces - A Complete Guide - Shafiqul AI, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://shafiqulai.github.io/blogs/blog_4.html>
107. Use FLUX, PyTorch, and Streamlit to Build an AI Image Generation App - Koyeb, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.koyeb.com/tutorials/use-flux-pytorch-and-streamlit-to-build-an-ai-image-generation-app>