# الشريحة 1: صفحة الغلاف (العنوان)

**مشروع تخرج بعنوان:**

# مرشدي الأكاديمي الذكي

## **(My AI Academic Advisor)**

**تطبيق معمارية الوكيل الذكي (Agentic RAG) لبناء نظام إرشاد أكاديمي استباقي وشخصي**

إعداد الطالب:

طارق معتز محيسن (4210380)

التخصص:

الهندسة المعلوماتية / هندسة الذكاء الصنعي وعلوم البيانات

بإشراف السادة:

الدكتور مهيب النقري

المهندسة آية الأسود

# الشريحة 2: فهرس المحتويات (Table of Contents)

1. **المقدمة والسياق (Introduction & Context)**
   * بيان المشكلة (The Problem)
   * الأهداف والغايات (The Objective)
2. **الأطروحة المركزية: محدودية RAG والحاجة إلى الوكيل (The Core Thesis)**
   * لماذا RAG التقليدي غير كافٍ؟
   * تقديم المنهجية: معمارية الوكيل الذكي (Agentic RAG)
3. **التصميم المعماري والهندسي (System Architecture & Engineering)**
   * المعمارية كضرورة: لماذا Docker والخدمات المصغرة؟ 1
   * مخطط النظام (Architecture Diagram)
   * المكدس التقني المفصل (Technology Stack)
4. **الأسس النظرية والدراسات المرجعية (Theoretical Foundations)**
   * المنهجيات المعتمدة في الاسترجاع والتنبؤ
5. **العمل المنجز: بناء "صندوق أدوات الوكيل" (Current Accomplishments)**
   * عرض حي للبنية التحتية (Backend Infrastructure)
   * الأدوات المنجزة (RAG, Graph, GPA Tools)
6. **التحدي الجوهري: الخصوصية والتحقق من البيانات (The Core Challenge)**
   * كيف يثق الطالب بالنظام؟ - حلول مقترحة
7. **الخطة الزمنية والآفاق المستقبلية (Roadmap & Future Work)**
8. **الخاتمة (Conclusion)**

# الشريحة 3: المقدمة وبيان المشكلة (Introduction & The Problem)

السياق:

يعتمد الطلاب والموظفون الإداريون في كلية الهندسة المعلوماتية حالياً على وثائق متفرقة (PDFs, Word) 1 ولوائح معقدة وأنظمة متعددة لإدارة المسار الأكاديمي. هذه العملية "تفاعلية" (Reactive)، غير فعالة، وعرضة للأخطاء البشرية.

**بيان المشكلة (The Problem):**

1. **تشتت المعرفة (Knowledge Silos):** اللوائح 1، الخطط الدراسية (القديمة والحديثة) 1، وتوصيف المقررات 1، كلها موجودة في ملفات منفصلة.
2. **الإرشاد السطحي (Superficial Guidance):** يواجه الطالب صعوبة في فهم "العلاقات" الخفية بين المواد والمهارات ومتطلبات سوق العمل.1
3. **التدخل المتأخر (Late Intervention):** يتم اكتشاف الطالب المتعثر (At-Risk) *بعد* صدور الإنذار الأكاديمي، عندما يكون التدخل صعباً.1
4. **العبء الإداري (Administrative Load):** يقضي المشرفون وقتاً طويلاً في الإجابة على أسئلة متكررة (FAQ) بدلاً من التركيز على الإرشاد الاستراتيجي.1

# الشريحة 4: الهدف الاستراتيجي (The Strategic Objective)

الهدف ليس بناء "برنامج دردشة" (Chatbot).

الهدف هو بناء "نظام إدراك أكاديمي" (Academic Cognitive System).1

**التحول من الإرشاد التفاعلي (Reactive) إلى الإرشاد الاستباقي والتنبؤي (Proactive & Predictive):**

* **الدقة (Accurate):** تقديم إجابات دقيقة 100% ومستندة إلى اللوائح الرسمية (Grounded Answers).1
* **التخصيص (Personalized):** تقديم إجابات تأخذ بعين الاعتبار خطة الطالب *الفعلية* ومعدله وتاريخه الأكاديمي.1
* **الاستباقية (Proactive):** توقع المشاكل قبل وقوعها (مثل التعثر أو تأخر التخرج) وتنبيه الطالب أو المشرف.1
* **الربط (Connected):** كشف "شجرة المهارات" (Skill Tree) و "المسارات المهنية" (Career Paths) المضمنة داخل الخطة الدراسية.1

# الشريحة 5: الأطروحة: محدودية RAG والحاجة إلى الوكيل الذكي

**الفرضية الأولى: RAG التقليدي غير كافٍ.**

الاسترجاع المعزز بالتوليد (RAG) التقليدي ممتاز للإجابة على الأسئلة من المستندات.1 ولكنه يفشل عندما يتطلب السؤال "منطقاً" أو "أدوات" متعددة.

(هنا تعرض صورة الدردشة التي أرسلتها 1 كدليل على بحثك)

دراسة حالة: "خلط الزيت بالماء" 1

عند اختبار RAG البسيط، لاحظنا "تضارباً معرفياً":

* **عند سؤاله عن "خطة ai":** يقوم RAG باسترجاع وثائق غير مفهومة (رموز) ويقدم إجابة خاطئة.1
* **عند سؤاله عن "مواد ai":** ينجح RAG ويسترجع بيانات صحيحة من المستندات (المصدر: RAG (ChromaDB)).1
* **عند سؤاله عن "تخصص التحكم":** يفشل RAG في العثور على أي شيء، فيقوم النموذج اللغوي بـ"الهلوسة" (المصدر: LLM (No RAG)).1

**الاستنتاج:** المرشد الأكاديمي ليس "أداة واحدة"، بل هو "صندوق أدوات". لا يمكننا استخدام "أداة RAG" للإجابة على سؤال يتعلق بـ "حساب المعدل".

# الشريحة 6: المنهجية المقترحة: معمارية الوكيل الذكي (Agentic RAG)

بما أن RAG التقليدي يفشل، فإن الحل هو **Agentic RAG**.2

ما هو الوكيل (Agent)؟

إنه "المدير" أو "العقل المدبر" (Orchestrator) الذي يمتلك "صندوق أدوات" (Toolbox). بدلاً من الإجابة مباشرة، يقوم الوكيل بالخطوات التالية 1:

1. **الفهم والتفكيك (Understand & Deconstruct):** يحلل سؤال الطالب المركب.
2. **اختيار الأداة (Tool Selection):** يقرر أي "أداة" من صندوق أدواته هي الأنسب للإجابة.1
3. **التنفيذ (Execution):** يستدعي الأداة (أو عدة أدوات بالتسلسل).
4. **التجميع والصياغة (Aggregation & Formulation):** يجمع النتائج من الأدوات ويستخدم النموذج اللغوي لصياغة إجابة واحدة شاملة.

(هنا تظهر صورة واجهة FastAPI التي أرسلتها)

لقد قمنا بالفعل ببناء واختبار "صندوق الأدوات" هذا بالكامل في الواجهة الخلفية (Backend).

# الشريحة 7: المعمارية كضرورة (Architectural Necessity)

مشروع بهذا التعقيد (وكيل، أدوات متعددة، قواعد بيانات مختلفة) لا يمكن بناؤه كملف واحد. الفشل في المعمارية هو فشل للمشروع.1

القرار الأول: المعمارية المنفصلة (Decoupled Architecture) 1

نفصل الواجهة (Streamlit) عن العقل المدبر (FastAPI).

* **لماذا؟** عمليات RAG واستدعاء النموذج اللغوي بطيئة. إذا كانت في نفس كود الواجهة، فإن الواجهة "ستتجمد" (Freeze) مع كل سؤال.1
* **الحل:** Streamlit 4 يعمل كـ "واجهة طرفية بسيطة" (Dumb Terminal) ترسل طلب HTTP، بينما يقوم FastAPI 6 بالعمل الثقيل في الخلفية بشكل غير متزامن.1

القرار الثاني: الخدمات المصغرة المُحَوّاة (Containerized Microservices) 1

* **لماذا؟** المشروع يعاني من "جحيم التبعيات" (Dependency Hell). Ollama يحتاج تعريفات GPU CUDA 12، و Scikit-learn يحتاج إصدارات numpy معينة، و spaCy يحتاج مكتبات نظام مختلفة.1
* **الحل:** Docker.14 نقوم بتغليف كل خدمة (مع كل تبعياتها من النظام والمكتبات) في حاوية معزولة ومستقلة.1

# الشريحة 8: مخطط المعمارية (System Architecture Diagram)

**[مخطط المعمارية هنا]**

(هذا هو المخطط الذي تضعه في الشريحة، وهو يمثل ما قمنا ببنائه):

1

* **صندوق (اليسار):** المستخدم (المتصفح)
* **سهم إلى:** الخدمة 1: الواجهة (Frontend)
  + الأداة: Streamlit 4
  + المنفذ: 8501
* **سهم إلى:** الخدمة 2: العقل المدبر (Backend / Agent)
  + الأداة: FastAPI 6
  + المنفذ: 8000
* **"العقل المدبر" (FastAPI) يتصل بأربع خدمات دعم:**
  + **سهم إلى (الخدمة 3): النموذج اللغوي (LLM)**
    - الأداة: Ollama (Llama 3 8B) 13
    - المنفذ: 11434
    - الميزة: NVIDIA GPU Access 12
  + **سهم إلى (الخدمة 4): ذاكرة RAG (الوثائق)**
    - الأداة: ChromaDB (Vector DB) 6
    - المنفذ: 8001
  + **سهم إلى (الخدمة 5): ذاكرة العلاقات (المعرفة)**
    - الأداة: Neo4j (Graph DB) 20
    - المنفذ: 7687
  + **سهم إلى (الخدمة 6 - مستقبلية): الذاكرة الدائمة (بيانات الطالب)**
    - الأداة: PostgreSQL (قاعدة بيانات علائقية)

# الشريحة 9: المكدس التقني المفصل (Technology Stack)

| **المجال** | **الأداة / التقنية** | **السبب (لماذا اخترنا هذه الأداة بالذات؟)** |
| --- | --- | --- |
| **التنسيق (Orchestration)** | **Docker / Docker Compose** | ضرورة حتمية لحل "جحيم التبعيات" (مثل CUDA, numpy) وضمان النشر الموثوق.15 |
| **الواجهة الخلفية (Backend)** | **FastAPI** | الأسرع في الأداء، يدعم (async) لفصل المنطق الثقيل عن الواجهة.6 |
| **الواجهة الأمامية (Frontend)** | **Streamlit** | الأسرع في بناء واجهات البيانات والدردشة التفاعلية.5 |
| **النموذج اللغوي (LLM)** | **Ollama (Llama 3 8B - GGUF)** | **الخصوصية:** يعمل محلياً 100% (بيانات الطلاب لا تغادر الخادم). **التكلفة:** مجاني (لا اشتراكات API). **الكفاءة:** GGUF هو التنسيق الأمثل للتشغيل المرن (CPU/GPU).25 |
| **تنسيق الذكاء الاصطناعي (AI Framework)** | **LangChain** | هو "الغراء" الصناعي لبناء سلاسل RAG و "الوكلاء" (Agents) القادرين على استخدام أدوات متعددة.30 |
| **ذاكرة RAG (الوثائق)** | **ChromaDB** | قاعدة بيانات متجهات مفتوحة المصدر، سريعة، ومتكاملة مع LangChain لفهرسة الوثائق (PDF/DOCX).6 |
| **ذاكرة العلاقات (المعرفة)** | **Neo4j** | RAG التقليدي يفشل في فهم "العلاقات". Neo4j ضروري لبناء "شجرة المهارات" والإجابة على أسئلة "المسار الوظيفي".20 |
| **التحليل والتنبؤ (Analysis)** | **Pandas / Scikit-learn** | المعيار الصناعي لتحليل الخطط 34 وبناء النماذج التنبؤية (مثل Random Forest).39 |
| **استخراج المعرفة (NLP)** | **spaCy / VADER** | spaCy لاستخراج المهارات (NER) 44 من توصيف المقررات. VADER لتحليل المشاعر (Sentiment Analysis).49 |

# الشريحة 10: الأسس النظرية والدراسات المرجعية (Theoretical Foundations)

هذا المشروع ليس مجرد تجميع أدوات، بل هو تطبيق لمنهجيات مثبتة:

1. **الاسترجاع الهرمي (Hierarchical RAG):**
   * لحل مشكلة RAG في الوثائق الأكاديمية، لا نستخدم التقطيع (Chunking) العادي.
   * نستخدم "التقطيع الهرمي" (Parent-Child Retriever).30
   * **كيف؟** نفهرس "قطع فرعية" (Child Chunks) صغيرة ودقيقة (للبحث)، ولكننا نسترجع "القطعة الأصل" (Parent Chunk) الكاملة (لإعطاء سياق كامل للـ LLM). هذا يوازن بين الدقة والسياق.
2. **الاسترجاع البياني (GraphRAG):**
   * للإجابة على أسئلة "العلاقات" (مثل: "ما هي المواد التي تفتحها مادة الخوارزميات؟").
   * نستخدم LangChain GraphCypherQAChain.20
   * **كيف؟** الـ LLM يترجم سؤال الطالب باللغة العربية إلى استعلام Cypher (لغة Neo4j)، يستعلم من قاعدة البيانات البيانية، ثم يترجم النتيجة إلى إجابة.
3. **التنبؤ الاستباقي (Proactive Prediction):**
   * بناءً على دراسة حالة جامعة ولاية جورجيا (GSU) 1، الهدف ليس إخبار الطالب "أنت ستفشل".
   * الهدف هو تطبيق نمط **"المرشد في الحلقة" (Advisor-in-the-Loop)**.1
   * **كيف؟** النموذج التنبؤي (Random Forest) 40 يحلل 800+ عامل خطر 1 ويرسل "تنبيهاً" (Alert) للمشرف البشري ("الطالب طارق سجل في مادة متقدمة وهو متعثر في متطلبها السابق").1

# الشريحة 11: العمل المنجز: بناء "صندوق أدوات الوكيل"

**الوضع الحالي: تم إنجاز البنية التحتية الكاملة (Backend Infrastructure) بنجاح.**

لقد قمنا ببناء وتشغيل واختبار "صندوق الأدوات" (Toolbox) الكامل الذي سيستخدمه "الوكيل الذكي" (Agent).

**(هنا تضع صورة واجهة FastAPI التي أرسلتها)**

| **الأداة (Endpoint)** | **الحالة** | **الوظيفة المنجزة (التي تم اختبارها)** |
| --- | --- | --- |
| **GET /** | **يعمل** | التحقق من أن "العقل المدبر" (FastAPI) قيد التشغيل. |
| **POST /ingest-pdfs** | **يعمل** | **أداة RAG:** تقرأ PDF و DOCX من مجلد /data وتفهرسها بنجاح في ChromaDB. |
| **POST /ingest-graph** | **يعمل** | **أداة Graph:** تقرأ بيانات التخصصات/المواد وتملأ Neo4j بنجاح.1 |
| **POST /analyze-plan** | **يعمل** | **أداة MVP:** تحسب المعدل، الساعات المتبقية، والمواد المتاحة للتسجيل.1 |
| **POST /simulate-gpa** | **يعمل** | **أداة MVP:** تحاكي المعدل المستقبلي بناءً على تقديرات متوقعة.1 |
| **POST /chat** | **يعمل** | **نقطة RAG الأساسية:** الواجهة (Streamlit) 1 تتصل بالعقل المدبر وتجيب من ChromaDB. |

# الشريحة 12: عرض حي / مقاطع منجزة (Live Demo / Clips)

**(هذه الشريحة لك لتضع فيها مقاطع الفيديو التي سجلتها)**

يجب أن تعرض للجنة مقطعين أو ثلاثة مقاطع فيديو قصيرة (Screen Recordings) تظهر "الأدوات" وهي تعمل:

* **المقطع 1: اختبار ingest-pdfs (دليل العمل 1)**
  + تصوير شاشة لصفحة FastAPI (localhost:8000/docs).
  + تنفيذ POST /ingest-pdfs.
  + إظهار رسالة النجاح: {"status": "success", "message": "Ingested and indexed 12 documents."}.
* **المقطع 2: اختبار analyze-plan (دليل العمل 2)**
  + تصوير شاشة لنفس الصفحة.
  + تنفيذ POST /analyze-plan مع إدخال JSON.
  + إظهار الرد JSON الصحيح الذي يحسب المعدل والمواد المتاحة.
* **المقطع 3: اختبار الواجهة (دليل العمل 3)**
  + تصوير شاشة لـ Streamlit (localhost:8501).
  + طرح سؤال من ملفات PDF (مثل: "ما هي اللائحة الداخلية؟").
  + إظهار الإجابة الصحيحة مع المصدر: (مصدر المعلومة: RAG (ChromaDB)).1

# الشريحة 13: التحدي الجوهري: خصوصية الطالب والتحقق من البيانات

بما أن النظام يتعامل مع بيانات حساسة (معدلات الطلاب وخططهم) 1، فإن "الثقة" و "التحقق" هي متطلبات أساسية، وليست خياراً.

**السؤال: كيف يثق الطالب "طارق" (كمستخدم) بالنظام للوصول إلى بياناته؟**

لقد قمنا بتصميم 3 حلول مقترحة لهذه المشكلة:

1. **الحل المعماري: "الصوامع المعرفية" (Cognitive Silos)**
   * نحن لا نخلط البيانات. بيانات الطالب الشخصية (معدلاته) **معزولة معمارياً** عن المعرفة العامة (لوائح PDF).
   * "الوكيل" (Agent) لا يصل مباشرة إلى قاعدة بيانات الطالب. بدلاً من ذلك، يستدعي "أداة" خاصة (مثل /get-my-plan) تتطلب مصادقة (Authentication) خاصة بالطالب.
2. **الحل الوظيفي: "RAG الوصفي" (Meta-Cognitive RAG)**
   * يتم تدريب الوكيل على الإجابة على "أسئلة حول البيانات" (Meta-Questions).
   * **مثال:** عندما يسأل الطالب "ما هي الخطة الدراسية التي تستخدمها لي؟"، يقوم الوكيل باستدعاء RAG ليس على المستندات، بل على **البيانات الوصفية (Metadata)** 1 المرتبطة بملف الطالب (ID: 4210380)، ليرد: "أنا أستخدم خطة 2022، إصدار هندسة الذكاء الاصطناعي".
3. **الحل عبر الواجهة: "الاستشهادات القابلة للتحقق" (Verifiable Citations)**
   * كما ظهر في اختبار الدردشة 1، كل إجابة يجب أن تحمل "مصدرها" بشكل واضح: (مصدر المعلومة: RAG (ChromaDB)) أو (مصدر المعلومة: أداة تحليل الخطة).
   * هذا يمنع "الصندوق الأسود" ويبني ثقة المستخدم، حيث يعرف الطالب ما إذا كانت الإجابة من لائحة رسمية أم استنتاج من النموذج اللغوي.

# الشريحة 14: الخطة الزمنية للإنجاز (Gantt Chart)

| **المرحلة** | **المهمة** | **الأسبوع 1-2** | **الأسبوع 3-4** | **الأسبوع 5-6** | **الأسبوع 7-8** | **الأسبوع 9** | **الأسبوع 10** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **1. الأساس (Foundation)** | إعداد المعمارية (Docker, 5 Services) | 🟩🟩🟩 |  |  |  |  |  |
|  | بناء أداة RAG (PDF/DOCX) |  | 🟩🟩🟩 |  |  |  |  |
|  | بناء أدوات MVP (الخطة/المعدل) |  |  | 🟩🟩🟩 |  |  |  |
|  | بناء أداة Graph (Neo4j) |  |  |  | 🟩🟩🟩 |  |  |
| **2. الذكاء (Intelligence)** | **بناء "الوكيل الموجه" (Agent Router)** |  |  |  | 🟧🟧🟧 |  |  |
|  | بناء أداة (GraphRAG) |  |  |  |  | 🟧🟧🟧 |  |
|  | بناء النموذج التنبؤي (Sentinel) |  |  |  |  | 🟧🟧🟧 |  |
| **3. النشر (Deployment)** | تصميم الواجهة الكاملة (Streamlit UI) |  |  |  |  |  | 🟨🟨🟨 |
|  | الاختبار الشامل والتوثيق |  |  |  |  |  | 🟨🟨🟨 |

**الحالة الحالية (مفتاح الألوان):** 🟩 (مُنجز وتم اختباره) | 🟧 (قيد التنفيذ) | 🟨 (مستقبلي)

# الشريحة 15: الآفاق المستقبلية (Future Work)

البنية التحتية جاهزة. العمل المستقبلي يركز على "الذكاء" و "الربط":

1. **بناء "الوكيل الموجه" (Agentic Router):**
   * **الخطوة الفورية:** ترقية نقطة النهاية /chat لاستخدام "وكيل" (Agent) من LangChain.2
   * **الهدف:** جعل الوكيل يقرر بذكاء: هل السؤال يتطلب أداة RAG أم أداة Graph أم أداة analyze-plan. هذا هو الحل لمشكلة "خلط الزيت بالماء" التي رأيناها.1
2. **تفعيل "أداة GraphRAG" الكاملة:**
   * لقد قمنا بفهرسة الرسم البياني (/ingest-graph). الخطوة التالية هي بناء الأداة التي تستعلم منه.
   * سنستخدم GraphCypherQAChain 20 لتمكين أسئلة مثل: "ما هي المهارات التي سأتعلمها في مسار الذكاء الاصطناعي؟".
3. **بناء "نظام الإنذار الاستباقي" (Proactive Sentinel):**
   * استخدام Scikit-learn 39 لتدريب نموذج RandomForestClassifier على بيانات الطلاب التاريخية (المجهولة الهوية).40
   * ربط "تحليل المشاعر" (VADER) 49 كـ "ميزة" (Feature) إضافية للنموذج لزيادة دقته.54
4. **تصميم "لوحة التحكم" (Dashboard) في Streamlit:**
   * الانتقال من واجهة الدردشة فقط إلى لوحة تحكم كاملة تعرض "شجرة المهارات" (Skill Tree) و "محاكي المعدل" (GPA Simulator) بشكل بصري.1

# الشريحة 16: الخاتمة (Conclusion)

"مرشدي الأكاديمي الذكي" هو أكثر من مجرد تطبيق؛ إنه **بنية تحتية معرفية**.

* **المساهمة الأساسية:**
  + تصميم وتنفيذ معمارية Agentic RAG معقدة، قابلة للتطوير، وموزعة على 5 خدمات Docker مستقلة.1
* **الإنجاز الحالي:**
  + تم بنجاح بناء واختبار "صندوق الأدوات" الكامل للوكيل، بما في ذلك فهرسة RAG (للوثائق) وفهرسة Graph (للعلاقات).
* **القيمة النهائية:**
  + تحويل الإرشاد الأكاديمي من عملية "تفاعلية" (Reactive) تعتمد على الذاكرة البشرية، إلى عملية "استباقية" (Proactive) و "تنبؤية" (Predictive) تعتمد على البيانات، مما يضمن الدقة للطالب ويوفر رؤى عميقة للمشرف الأكاديمي.

# الشريحة 17: شكراً لحسن استماعكم

طارق معتز محيسن

tariq.mohaisen@email.com

**(أسئلة وأجوبة)**

#### المصادر التي تم الاقتباس منها

1. بناء مرشد أكاديمي ذكي باستخدام Docker.docx
2. 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/>
3. Intro to GraphRAG, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://graphrag.com/concepts/intro-to-graphrag/>
4. 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>
5. How to Build an Instant Machine Learning Web Application with Streamlit and FastAPI, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://developer.nvidia.com/blog/how-to-build-an-instant-machine-learning-web-application-with-streamlit-and-fastapi/>
6. Build a Modular RAG Chatbot with FastAPI, Streamlit & ChromaDB - YouTube, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.youtube.com/watch?v=TxtK6NUUklQ>
7. My Experience Building A FastAPI + Streamlit App - Pybites, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://pybit.es/articles/my-experience-building-a-fastapi-streamlit-app/>
8. Building a Basic RAG App with LangGraph, FastAPI & ChromaDB | by Prashant Suthar, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://prashant1879.medium.com/building-a-basic-rag-app-with-langgraph-fastapi-chromadb-668c7454d035>
9. Building a RAG Pipeline with FastAPI, Haystack, and ChromaDB for URLs in Python, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@pwaykos1/building-a-rag-pipeline-with-fastapi-haystack-and-chromadb-for-urls-in-python-631575f3888b>
10. Building a Production-Ready RAG Chatbot with FastAPI and LangChain, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://blog.futuresmart.ai/building-a-production-ready-rag-chatbot-with-fastapi-and-langchain>
11. How to build frontend for RAG application? [need help] : r/LangChain - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/LangChain/comments/1eub0cb/how_to_build_frontend_for_rag_application_need/>
12. Ollama Docker image, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://hub.docker.com/r/ollama/ollama>
13. Ollama is now available as an official Docker image, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://ollama.com/blog/ollama-is-now-available-as-an-official-docker-image>
14. Docker Compose Quickstart - Docker Docs, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.docker.com/compose/gettingstarted/>
15. A Beginner's Guide to Docker Compose | Better Stack Community, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://betterstack.com/community/guides/scaling-docker/docker-compose-getting-started/>
16. A Simple Guide to Docker Compose & Multi-Container Applications - DEV Community, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://dev.to/mukhilpadmanabhan/a-simple-guide-to-docker-compose-multi-container-applications-5e0g>
17. Docker Compose Guide: Simplify Multi-Container Development - DataCamp, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.datacamp.com/tutorial/docker-compose-guide>
18. Run LLMs locally or in Docker with Ollama & Ollama-WebUI | by Namrata D - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://namrata23.medium.com/run-llms-locally-or-in-docker-with-ollama-ollama-webui-379029060324>
19. mythrantic/ollama-docker: Welcome to the Ollama Docker Compose Setup! This project simplifies the deployment of Ollama using Docker Compose, making it easy to run Ollama with all its dependencies in a containerized environment - GitHub, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://github.com/mythrantic/ollama-docker>
20. Neo4j - Docs by LangChain, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://docs.langchain.com/oss/python/integrations/graphs/neo4j_cypher>
21. LangChain Cypher Search: Tips & Tricks - Neo4j, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://neo4j.com/blog/developer/langchain-cypher-search-tips-tricks/>
22. Integrating Neo4j with Langchain for Graph-Based Question Answering | by Vishnu Singh, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@vishnu.19je0919/integrating-neo4j-with-langchain-for-graph-based-question-answering-88fdb2cd7d8c>
23. Enhanced QA Integrating Unstructured Knowledge Graph Using Neo4j and LangChain, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://neo4j.com/blog/developer/unstructured-knowledge-graph-neo4j-langchain/>
24. Implementing advanced RAG strategies with Neo4j - LangChain Blog, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://blog.langchain.com/implementing-advanced-retrieval-rag-strategies-with-neo4j/>
25. A Visual Guide to Quantization - Maarten Grootendorst, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.maartengrootendorst.com/blog/quantization/>
26. Simplifying Quantization in LLMs: GGUF, GPTQ, AWQ and More | by Anand Sahu | Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@anand_sahu/simplifying-quantization-in-llms-gguf-gptq-awq-and-more-4c472722e28c>
27. 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>
28. Which Quantization Method is Right for You? (GPTQ vs. GGUF vs. AWQ) - YouTube, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.youtube.com/watch?v=mNE_d-C82lI>
29. For those who don't know what different model formats (GGUF, GPTQ, AWQ, EXL2, etc.) mean ↓ : r/LocalLLaMA - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/LocalLLaMA/comments/1ayd4xr/for_those_who_dont_know_what_different_model/>
30. LangChain: ParentDocumentRetriever Quick Guide - Kaggle, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.kaggle.com/code/ksmooi/langchain-parentdocumentretriever-quick-guide>
31. The Beauty of Parent-Child Chunking. Graph RAG Was Too Slow for Production, So This Parent-Child RAG System was useful - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/Rag/comments/1mtcvs7/the_beauty_of_parentchild_chunking_graph_rag_was/>
32. Parent-Child Chunking in LangChain for Advanced RAG | by Seahorse - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@seahorse.technologies.sl/parent-child-chunking-in-langchain-for-advanced-rag-e7c37171995a>
33. LangChain Parent-Child Retriever for better RAG - YouTube, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.youtube.com/watch?v=wSi0fxkH6e0>
34. Advanced Feature Engineering with Pandas - GeeksforGeeks, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.geeksforgeeks.org/machine-learning/advanced-feature-engineering-with-pandas/>
35. Feature Engineering with Pandas | CodeSignal Learn, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://codesignal.com/learn/courses/cleaning-and-transforming-data-with-pandas/lessons/feature-engineering-with-pandas>
36. Learn Feature Engineering | Kaggle, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.kaggle.com/learn/feature-engineering>
37. 7 of the Most Used Feature Engineering Techniques | Towards Data Science, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://towardsdatascience.com/7-of-the-most-used-feature-engineering-techniques-bcc50f48474d/>
38. Feature Engineering and Model Building with Python and Pandas | by Durga Gadiraju | itversity | Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/itversity/feature-engineering-and-model-building-with-python-and-pandas-196edaefc9ac>
39. Random Forest Classification in Python With Scikit-Learn: Step-by-Step Guide (with Code Examples) | DataCamp, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.datacamp.com/tutorial/random-forests-classifier-python>
40. RandomForestClassifier — scikit-learn 1.7.2 documentation, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
41. 1.11. Ensembles: Gradient boosting, random forests, bagging, voting, stacking - Scikit-learn, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://scikit-learn.org/stable/modules/ensemble.html>
42. Random Forest Algorithm Explained with Python and scikit-learn - YouTube, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.youtube.com/watch?v=_QuGM_FW9eo>
43. Using Random Forests in Python with Scikit-Learn | Oxford Protein Informatics Group, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.blopig.com/blog/2017/07/using-random-forests-in-python-with-scikit-learn/>
44. spaCy 101: Everything you need to know · spaCy Usage Documentation, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://spacy.io/usage/spacy-101>
45. Named Entity Recognition (NER) in Python with Spacy - Analytics Vidhya, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.analyticsvidhya.com/blog/2021/06/nlp-application-named-entity-recognition-ner-in-python-with-spacy/>
46. Named Entity Recognition -How to extract skill entities from resumes using spaCy - Medium, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/hr-ai/named-entity-recognition-how-to-extract-skill-entities-from-resumes-using-spacy-865476b5771e>
47. Skill extraction from resumes : r/LanguageTechnology - Reddit, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.reddit.com/r/LanguageTechnology/comments/uykvx0/skill_extraction_from_resumes/>
48. How to Extract Information from Text with SpaCy - YouTube, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.youtube.com/watch?v=1S8icpu9dX0>
49. Sentiment Analysis using VADER - Using Python - GeeksforGeeks, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.geeksforgeeks.org/python/python-sentiment-analysis-using-vader/>
50. Sentiment Analysis Using VADER - Analytics Vidhya, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.analyticsvidhya.com/blog/2022/10/sentiment-analysis-using-vader/>
51. Vader: A Comprehensive Guide to Sentiment Analysis in Python | by Lavanya Geetha, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://medium.com/@rslavanyageetha/vader-a-comprehensive-guide-to-sentiment-analysis-in-python-c4f1868b0d2e>
52. Sentiment Analysis in Python with TextBlob and VADER Sentiment (also Dash p.6), تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.youtube.com/watch?v=qTyj2R-wcks>
53. Sentiment Analysis for Exploratory Data Analysis - Programming Historian, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://programminghistorian.org/en/lessons/sentiment-analysis>
54. What is Sentiment Analysis? - Amazon AWS, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://aws.amazon.com/what-is/sentiment-analysis/>
55. Using Text Sentiment as feature in machine learning model? - Stack Overflow, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://stackoverflow.com/questions/46253404/using-text-sentiment-as-feature-in-machine-learning-model>
56. Using Machine Learning for Sentiment Analysis: a Deep Dive | DataRobot Blog, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.datarobot.com/blog/using-machine-learning-for-sentiment-analysis-a-deep-dive/>
57. What Is Sentiment Analysis? - IBM, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://www.ibm.com/think/topics/sentiment-analysis>
58. A complete guide to Sentiment Analysis approaches with AI - Thematic, تم الوصول بتاريخ ‎نوفمبر 11, 2025، <https://getthematic.com/sentiment-analysis>