ЛАБОРАТОРНА РОБОТА № 3. РОЗРОБКА ТА РОЗШИРЕННЯ ФУНКЦІОНАЛЬНОСТІ AI-АГЕНТА З ВИКОРИСТАННЯМ LANGCHAIN ТА LANGGRAPH

Мета: Навчитися аналізувати існуючий код AI-агента, розробленого на базі LangChain. Навчитись працювати з ключовими компонентами LangChain: LLM, Prompt Templates, Output Parsers, Chains (LCEL). Навчитись створювати та інтегрувати власні інструменти (Tools) для агента. Закріпити знання про архітектуру ReAct-агентів, важливість системних промптів та автономні робочі процеси. Отримати практичний досвід модифікації та розширення існуючої кодової бази.

Теоретичні відомості

AI-агент – це автономна система, що використовує велику мовну модель (LLM) як центральний «мозок» для прийняття рішень та виконання дій. Ефективність агента залежить не тільки від потужності LLM, але й від його здатності взаємодіяти із зовнішнім світом. Цю взаємодію забезпечують **інструменти (Tools)** – функції, які агент може викликати для доступу до даних, сервісів або виконання операцій, що недоступні самій LLM (наприклад, пошук в інтернеті, робота з файлами, доступ до API).

LangChain – провідний фреймворк для розробки додатків на базі LLM. Він виступає як потужний оркестратор, що дозволяє поєднувати мовні моделі із зовнішніми джерелами даних та сервісами. Фундаментальна ідея фреймворку полягає в модульній композиції. Замість написання монолітного коду розробники використовують LangChain Expression Language (LCEL), щоб створювати потоки обробки даних, з'єднуючи окремі компоненти, такі як промпти, моделі та парсери виводу, в єдиний ланцюжок. Саме LangChain надає готові реалізації для агентів, таких як ReAct, та спрощує процес створення та інтеграції інструментів, перетворюючи звичайні Python-функції на потужні інструменти, доступні для LLM. Окрім цього, фреймворк містить інструменти для роботи з пам'яттю, що дозволяє агентам пам'ятати попередні взаємодії, а також механізми для роботи з документами та базами даних для побудови систем доповненої генерації (RAG).

ЗАВДАННЯ ДЛЯ ВСІХ ВАРІАНТІВ

Крок 1. Ознайомлення з проєктом та налаштування середовища

1. Ознайомтеся з наданою кодовою базою (файли *agent.py*, *config.py*, *routes/analysis.py* тощо). Проаналізуйте, як створюється агент (*create_agent_executor*), які інструменти (*TavilySearch*) йому надано за замовчуванням, та як він інтегрований у FastAPI-додаток.
2. Налаштуйте робоче середовище: встановіть усі залежності з *requirements.txt*, створіть *.env* файл та надайте необхідні API-ключі (Google, Tavily, LangGraph).
3. Запустіть проєкт та переконайтесь, що він працює. Протестуйте вкладку «AI Agent», поставивши йому запитання, що вимагає пошуку в інтернеті.

4. **Придумайте та опишіть ідею для нового інструмента, який був би корисним для «AI-асистента».**

   Інструмент має виконувати дію, яку LLM не може зробити самостійно (наприклад, точний розрахунок, звернення до специфічних даних, запис у файл).

5. **Опишіть ідею для AI-агента, який би використовував існуючий інструмент пошуку та ваш новий інструмент для вирішення комплексної задачі.**

   - *Дайте агенту назву (напр., «AI-Планувальник Погоди», «Науковий Асистент»).*
   - *Сформулюйте для нього конкретну, високорівневу мету, яка демонструє спільну роботу інструментів.*
   - *Опишіть, як саме агент буде використовувати інструмент для досягнення мети.*

Крок 2. Технічна реалізація

1. **Створіть новий Python-файл для вашого скрипта (напр., *lab_main.py*).**
2. **Ваш скрипт повинен містити:**

   - *Ініціалізацію LLM (можна запозичити логіку з config.py).*
   - *Створення вашого кастомного інструмента. Напишіть Python-функцію для вашого інструмента та перетворіть її на інструмент для LangChain за допомогою декоратора @tool. Не забудьте додати детальний опис (docstring) до функції – агент буде використовувати цей опис, щоб зрозуміти, коли викликати інструмент.*

- *Ініціалізацію стандартних інструментів (напр., TavilySearch).*
- *Створення agent_executor з повним списком інструментів (стандартні + ваш кастомний).*
- *Формулювання конкретного завдання (task), яке агент має виконати автономно, використовуючи ваші інструменти.*
- *Асинхронний запуск агента та вивід результату.*
- *За бажанням, робіть будь-які зміни у поточній архітектурі та змісті файлів.*

Крок 3. Підготовка звіту

Оформіть звіт за наступною структурою:

1. *Тема, мета роботи.*
2. *Опис ідеї та завдання для агента: назва агента, його мета та сценарій використання, опис створеного інструмента.*
3. *Хід роботи: повний код скрипта з коментарями та описом ключових частин (створення інструмента, конфігурація агента).*
4. *Результати виконання: скріншот консолі з логами роботи агента (verbose=True), де видно його «думки» та виклики інструментів. Якщо агент створює артефакт (файл), додати його вміст.*
5. *Висновки: що вдалося реалізувати, з якими труднощами зіткнулися (напр., агент неправильно викликав інструмент, проблеми з промптом), та як можна покращити агента.*

ХІД РОБОТИ

Частина 1. Налаштування середовища та запуск базового проєкту.

Перед тим, як створювати власні компоненти, необхідно розгорнути та протестувати базову версію додатку, що дозволить вам ознайомитися з його архітектурою та переконатися, що ваше середовище налаштовано правильно.

Огляд технологічного стеку

- ***FastAPI:*** *бекенд-фреймворк. Обраний за швидкість, сучасність та автоматичну генерацію документації.*
- ***LangChain:*** *оркестрація AI-операцій. Дозволяє легко створювати ланцюжки AI-операцій, використовувати шаблони, отримувати структурований вивід (JSON) та обробляти помилки.*
- ***Google Gemini:*** *модель штучного інтелекту. Обрана через доступність, високу продуктивність та мультимодальні можливості (текст + зображення).*
- ***Tavily AI:*** *пошуковий сервіс для досліджень. Надає якісні результати пошуку, оптимізовані для AI-агентів.*
- ***Pydantic:*** *валідація даних. Забезпечує безпеку типів та автоматичну валідацію вхідних та вихідних даних.*
- ***Python 3.11+:*** *мова програмування.*

Крок 1. Клонування репозиторію та створення структури

Спочатку завантажте проєкт з GitHub.

```
git clone https://github.com/F1-bot/langchain_sample.git
cd construction-ai/backend
```

Крок 2. Створення та активація віртуального середовища

Ізолюємо залежності вашого проєкту від системних.

```
# Створення середовища
python -m venv venv
```

```
# Windows:
venv\Scripts\activate
# Mac/Linux:
source venv/bin/activate
```

Крок 3. Встановлення залежностей

Встановіть усі необхідні бібліотеки з файлу requirements.txt.

```
pip install --upgrade pip
pip install -r requirements.txt
```

Крок 4. Налаштування змінних середовища

Змінні середовища дозволяють зберігати конфіденційну інформацію окремо від коду. Створіть файл .env у кореневій папці backend/ та заповніть його за зразком з .env.example.

Вміст файлу *.env*:

```
GOOGLE_API_KEY=AIzaSy...ваш_ключ_тут
TAVILY_API_KEY=tvly-...ваш_ключ_тут
LANGCHAIN_TRACING_V2=true
LANGCHAIN_ENDPOINT=https://api.smith.langchain.com
LANGCHAIN_API_KEY=lsv2_pt_...ваш_ключ_langsmith_тут
HOST=0.0.0.0
PORT=8000
CORS_ORIGINS=http://localhost:3000,http://127.0.0.1:3000
GEMINI_MODEL=gemini-pro
TEMPERATURE=0.7
MAX_TOKENS=2048
```

Примітка:

TAVILY_API_KEY – https://www.tavily.com/

GOOGLE_API_KEY – https://aistudio.google.com/

LANGCHAIN_API_KEY – https://smith.langchain.com/

Крок 5. Запуск додатку

Переконайтесь, що ви перебуваєте в директорії backend і ваше віртуальне середовище активоване.

```
uvicorn app.main:app --reload
```

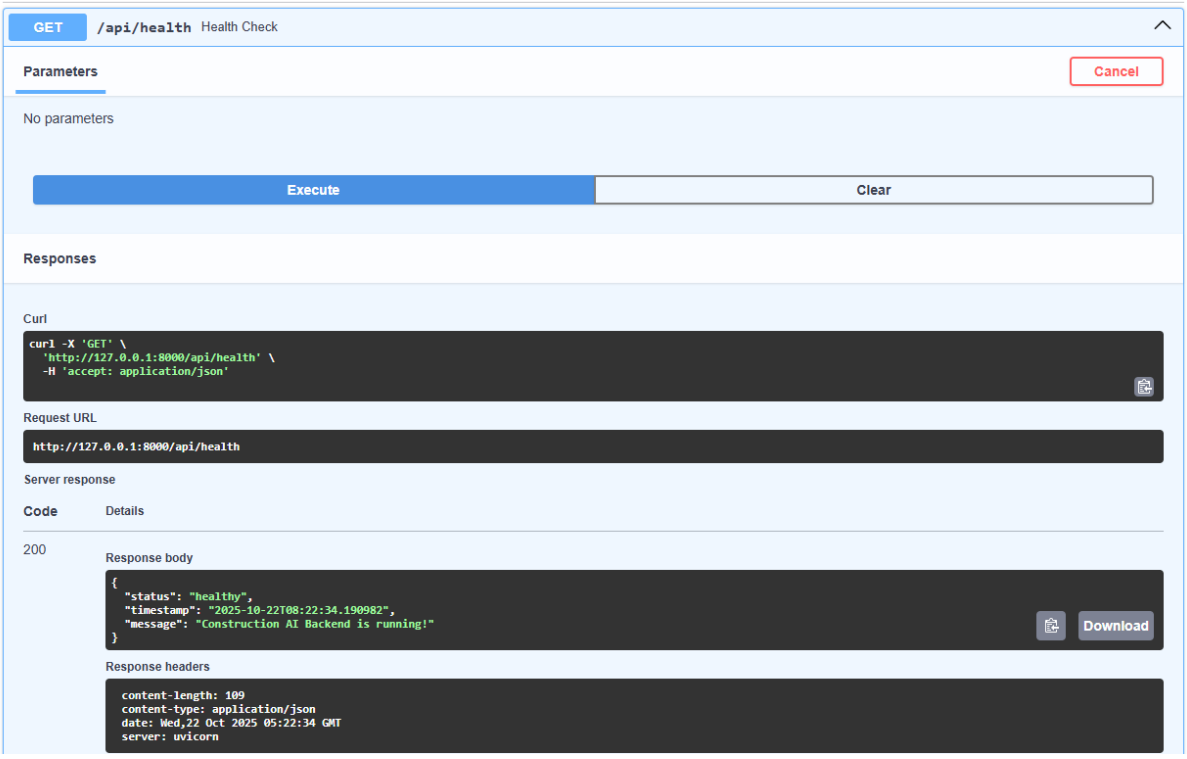
--reload автоматично перезапускає сервер при зміні коду. Ви повинні побачити повідомлення: uvicorn running on http://127.0.0.1:8000.

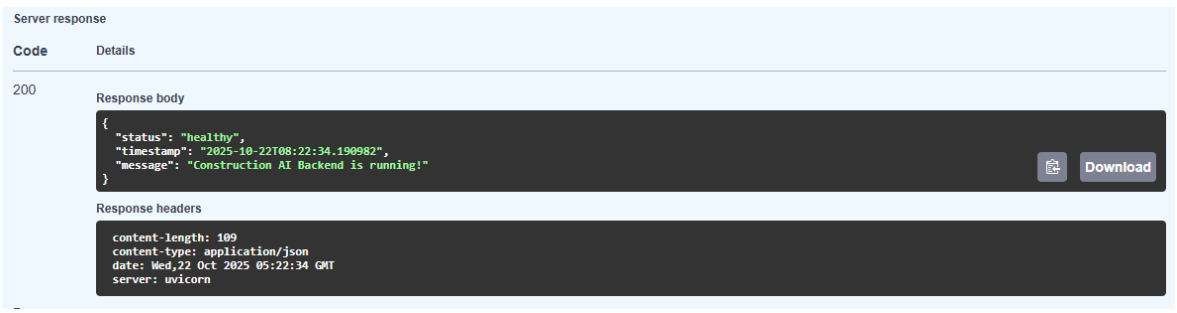
Крок 6. Тестування API через Swagger UI

FastAPI автоматично генерує інтерактивну документацію. Відкрийте у браузері: http://localhost:8000/docs.

Тест 1. Health Check (GET /api/health)

1. *Знайдіть секцію Health, розкрийте її.*
2. *Натисніть «Try it out», а потім «Execute».*
3. *Ви повинні отримати відповідь 200 OK з повідомленням «Construction AI Backend is running!». Це означає, що сервер працює коректно.*



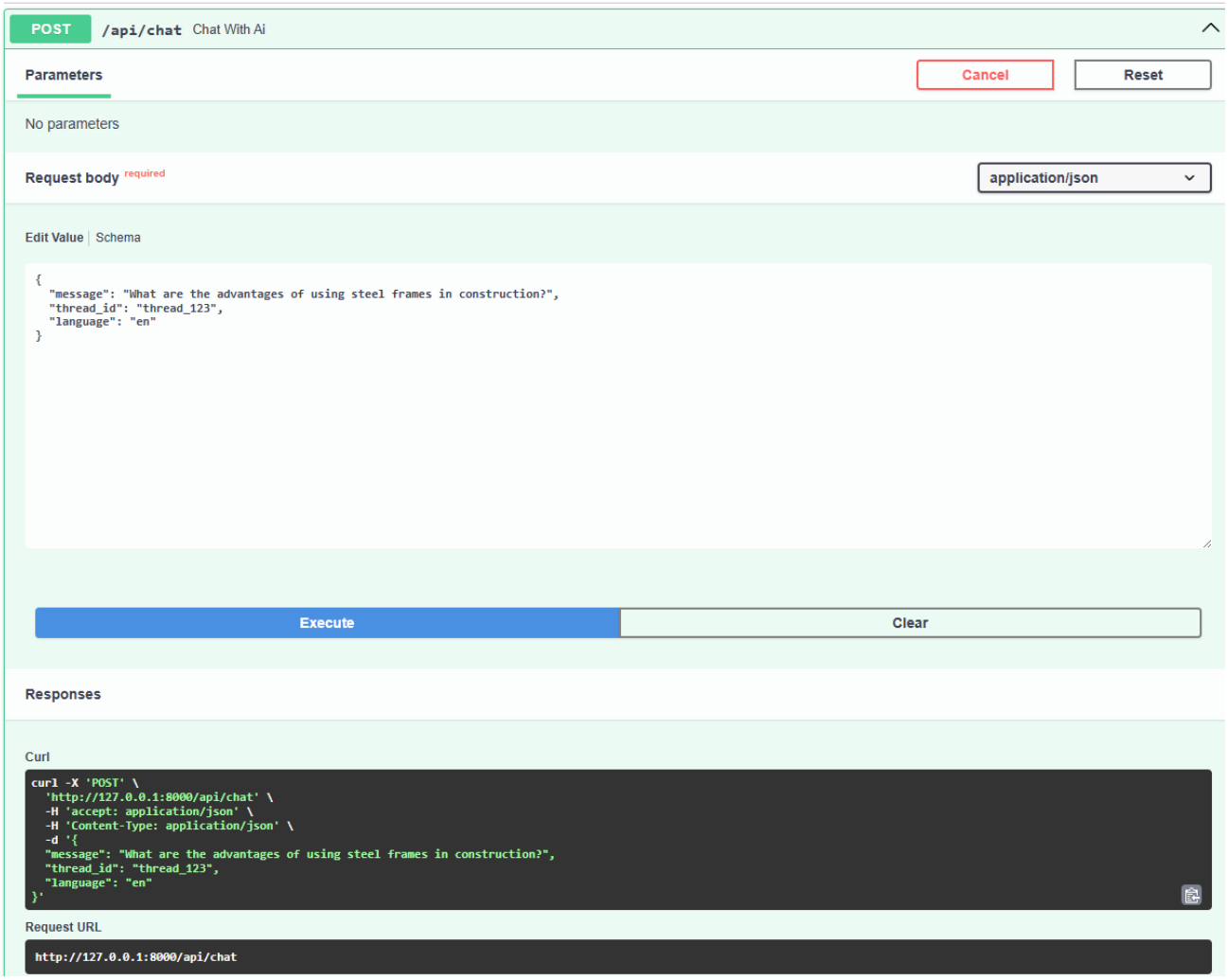


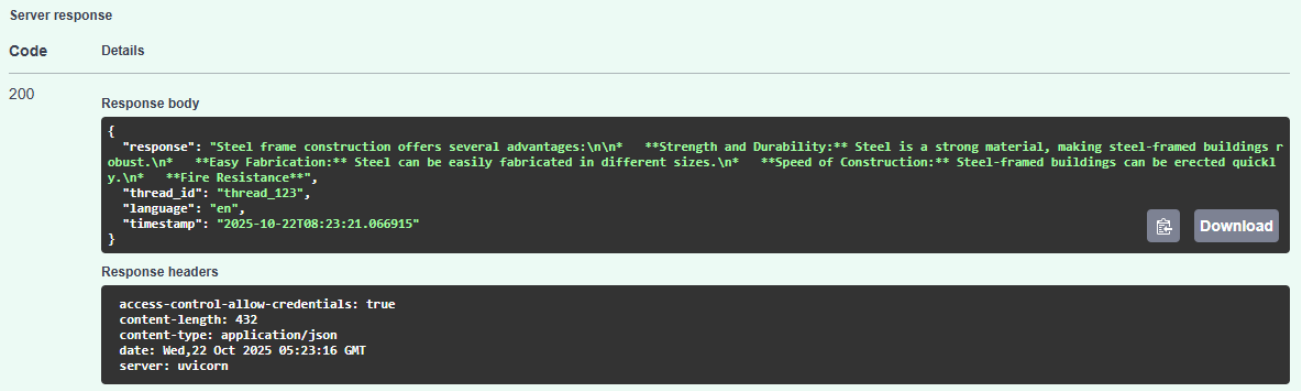
Тест 2. Чат з AI (POST /api/chat)

1. *Знайдіть секцію Analysis, розкрийте ендпоінт POST /api/chat.*
2. *Натисніть «Try it out».*
3. *У тілі запиту (Request body) введіть будівельне питання. Наприклад:*

```
{
  "message": " What are the advantages of using steel frames in
construction?",
  "thread_id": "thread_123",
  "language": "en"
}
```

4. *Натисніть «Execute» та перегляньте відповідь агента.*





Тест 3. Аналіз тексту (POST /api/analyze-text)

Цей ендпоінт аналізує текст будівельного документа.

1. *Розкрийте POST /api/analyze-text та натисніть «Try it out».*
2. *Введіть зразок звіту:*

```
{
  "text": "Inspection report dated 20.10.2025. Object: Skyline residential complex. Cracks up to 2 mm wide were found in the foundation of section B. Reinforcement bars are exposed in column 3-A.",
  "context": "Scheduled inspection of load-bearing structures",
  "language": "en"
}
```

3. *Спостерігайте, як AI повертає структуровану відповідь з полями summary, key_findings, recommendations.*

POST /api/analyze-text Analyze Text

Parameters | Cancel | Reset

No parameters

Request body required | application/json

Edit Value | Schema

```
{
  "text": "Inspection report dated 20.10.2025. Object: Skyline residential complex. Cracks up to 2 mm wide were found in the foundation of section B. Reinforcement bars are exposed in column 3-A.",
  "context": "Scheduled inspection of load-bearing structures",
  "language": "en"
}
```

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/api/analyze-text' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "text": "Inspection report dated 20.10.2025. Object: Skyline residential complex. Cracks up to 2 mm wide were found in the foundation of section B. Reinforcement bars are exp
  "context": "Scheduled inspection of load-bearing structures",
  "language": "en"
}'
```

Request URL

```
http://127.0.0.1:8000/api/analyze-text
```

Server response

Code	Details
200	Response body

```
{
  "summary": "Inspection report from 20.10.2025 for the Skyline residential complex, section B, reveals structural concerns regarding the foundation and a specifi
c column.",
  "key_findings": [
    "Cracks up to 2mm wide were found in the foundation of section B.",
    "Reinforcement bars are exposed in column 3-A."
  ],
  "recommendations": [
    "Engage a structural engineer to assess the severity and potential causes of the foundation cracks.",
    "A structural engineer should assess the condition of the exposed reinforcement bars in column 3-A and determine the necessary repair strategy.",
    "Conduct a comprehensive non-destructive testing (NDT) assessment of the affected areas to determine the extent of the damage and potential for further deteri
oration."
  ],
  "next_steps": [
    "Immediately consult with a qualified structural engineer to evaluate the findings.",
    "Implement temporary shoring or support measures if deemed necessary by the structural engineer.",
    "Develop a detailed repair plan based on the structural engineer's assessment.",
    "Schedule and execute the required repairs under the supervision of qualified professionals.",
    "Following repairs, conduct a follow-up inspection to ensure the effectiveness of the remediation efforts."
  ],
  "disclaimer": "This analysis is for informational purposes only. Always consult qualified construction professionals for advice.",
  "language": "en",
  "timestamp": "2025-10-22T08:24:41.245920"
}
```

Download

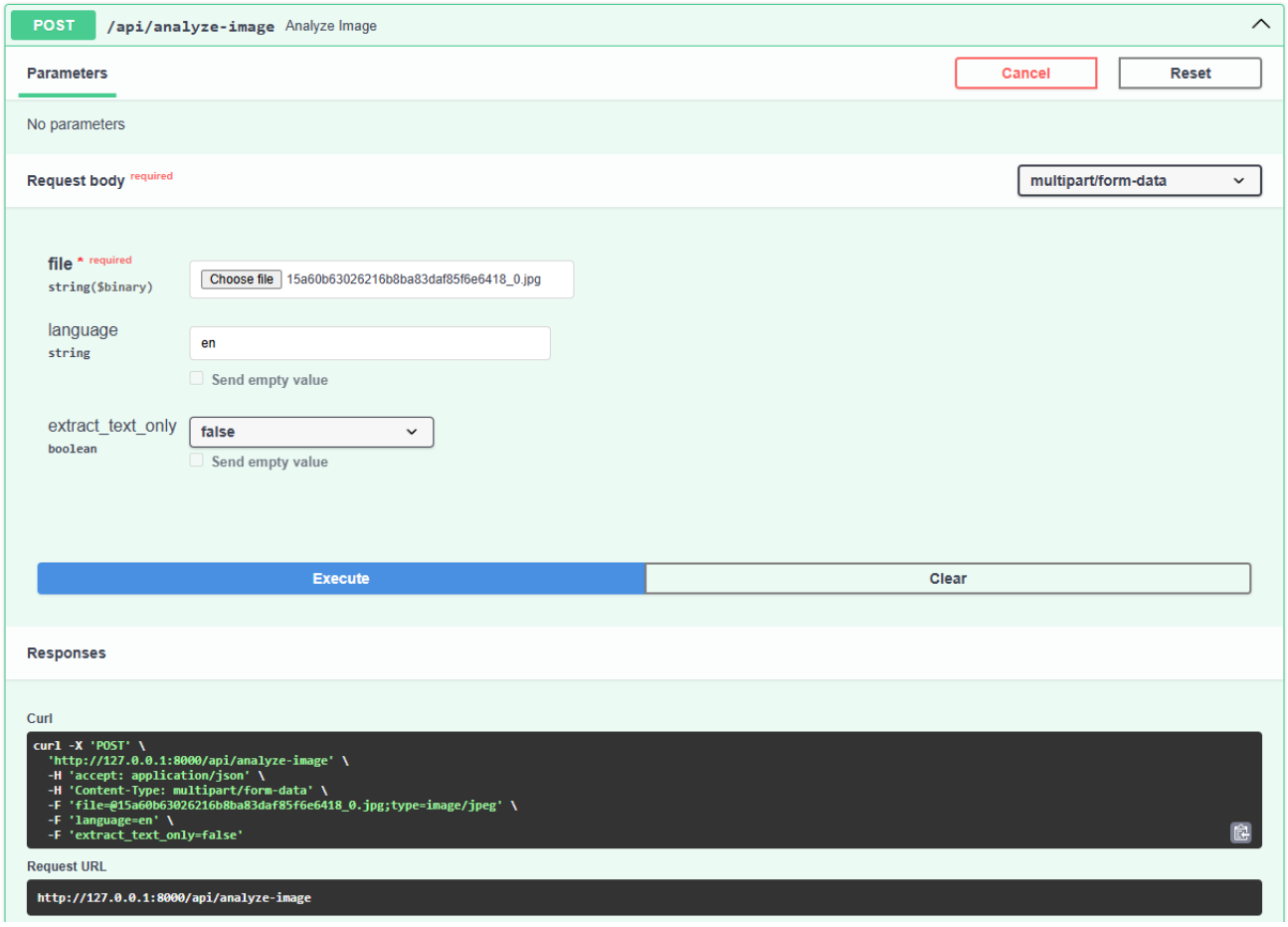
Response headers

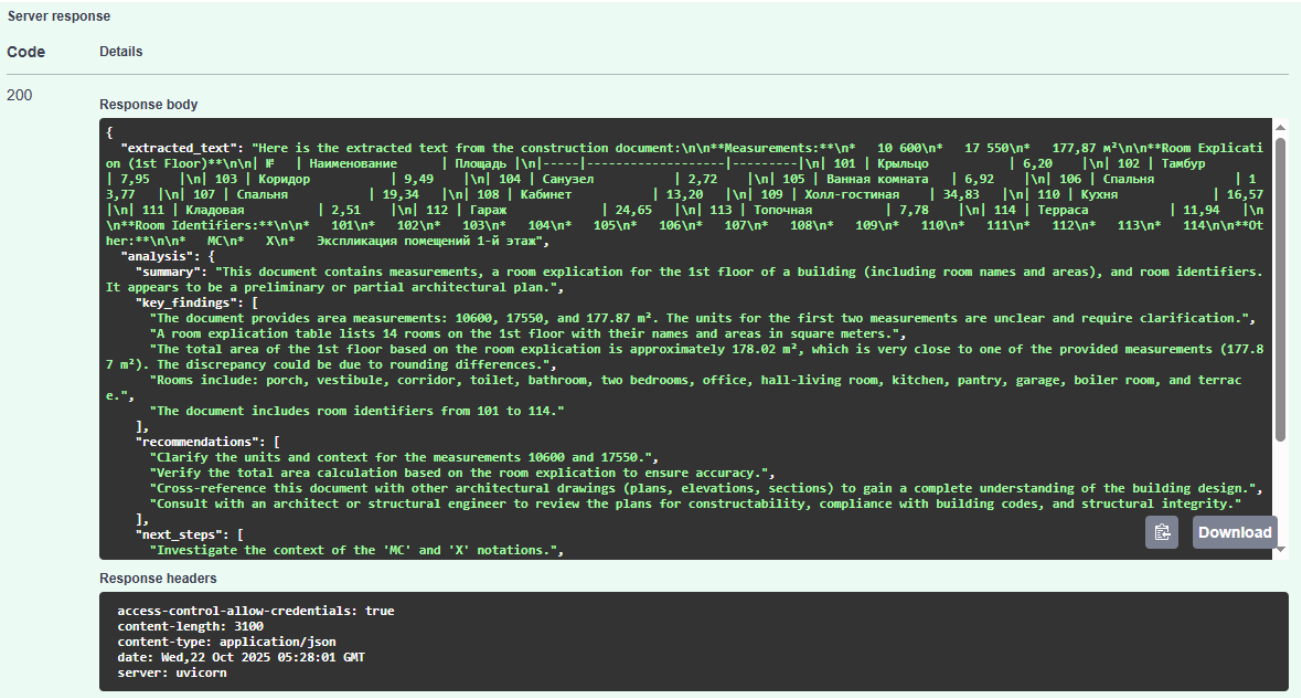
```
access-control-allow-credentials: true
content-length: 1403
content-type: application/json
date: Wed,22 Oct 2025 05:24:37 GMT
server: uvicorn
```

Тест 4. Аналіз зображення (POST /api/analyze-image)

Тут Gemini Vision вилучає текст з зображення (креслення, звіту) та аналізує його.

1. *Розкрийте POST /api/analyze-image та натисніть «Try it out».*
2. *За допомогою кнопки «Choose File» завантажте зображення (наприклад, фотографію частини плану або специфікації).*
3. *Натисніть «Execute».*
4. *У відповіді ви отримаєте як вилучений текст (extracted_text), так і його повний аналіз (analysis).*





Тест 5. Дослідження (POST /api/research)

Цей ендпоінт шукає інформацію у надійних будівельних джерелах.

1. *Розкрийте POST /api/research та натисніть «Try it out».*
2. *Введіть пошуковий запит:*

```
{
  "query": "latest technologies in energy-efficient construction",
  "max_results": 3,
  "language": "en"
}
```

3. *У відповіді ви отримаєте список джерел та коротке резюме (summary), згенероване AI.*

POST `/api/research` Search Construction Research

Parameters | Cancel | Reset

No parameters

Request body required | application/json

Edit Value | Schema

```
{
  "query": "latest technologies in energy-efficient construction",
  "max_results": 3,
  "language": "en"
}
```

Execute | Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/api/research' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "query": "latest technologies in energy-efficient construction",
  "max_results": 3,
  "language": "en"
}'
```

Request URL

```
http://127.0.0.1:8000/api/research
```

Server response

Code	Details
200	Response body


```
{
  "query": "latest technologies in energy-efficient construction",
  "results": [
    {
      "title": "Construction Technology is Reshaping the Industry",
      "url": "https://www.constructconnect.com/blog/technology-reshaping-construction-industry?__hstc=251652889.2f3f33a24b44870ec4a577029c49e44b.1757548800101.1757548800102.1757548800103.1&__hssc=251652889.1.17575488001048__hsfp=2825657416&hsCtaTracking=028f85be-0ff9-4baa-81b7-514a9079ce61%7C2e438af9-8fdc-4292-a884-1a275b158b86",
      "content": "The point is, advancements in new construction technology have always driven construction forward, so it's odd that so many companies are slow to adopt new construction technologies. We're able to build stronger, taller, and more energy efficient structures. Technology has made construction sites safer and workers more efficient. It has allowed us to increase productivity, improve collaboration, and tackle more complex projects.\n\n## What is Construction Technology? [...] Today, new technologies ",
      "score": 0.43065065
    },
    {
      "title": "6 Steps to Creating and Increasing Building Efficiency",
      "url": "https://www.buildings.com/resiliency-sustainability/energy-water-efficiency/article/55294612/6-steps-to-creating-and-increasing-building-efficiency",
      "content": "There is a wealth of technology available today that can help save time and money and deliver insights based on real-time data to improve overall building efficiency. Improved use of technology may be as simple as equipping operators with mobile capabilities or opting for a fully integrated building operations platform. These technologies are constantly evolving, so to maintain optimum efficiency it is essential for building operators to keep their technology stack up to date.",
      "score": 0.3454811
    },
    {
      "title": "7 Passive Cooling Strategies for Energy-Efficient Buildings",
      "url": "https://www.buildings.com/building-systems-om/article/55277269/7-passive-cooling-strategies-for-energy-efficient-buildings",
      "content": "Managing a building is a constant opportunity, primarily for energy-efficient improvements like passive cooling. These strategies involve num
```


Download

Response headers

```
access-control-allow-credentials: true
content-length: 2962
content-type: application/json
date: Wed,22 Oct 2025 05:29:09 GMT
server: uvicorn
```

Крок 7. Аналіз трасування (Trace) в LangSmith

FastAPI автоматично генерує інтерактивну документацію. Відкрийте у браузері: http://localhost:8000/docs.

1. *Перейдіть на сайт smith.langchain.com.*
2. *Увійдіть, використовуючи той самий акаунт, для якого ви генерували LANGCHAIN_API_KEY.*
3. *Ви побачите проєкт, який був автоматично створений для ваших запусків.*
4. *У списку проєктів ви побачите останні запуски (runs). Кожен ваш запит до API (наприклад, через Swagger) відповідає одному запису в цьому списку.*
5. *Знайдіть запис, що відповідає вашому останньому тесту (наприклад, запит до /api/chat).*
6. *Натисніть на запис, щоб відкрити детальне трасування. Ви побачите ієрархічний вигляд усіх операцій, що відбулися для генерації відповіді.*
7. *Зверніть увагу на ключові етапи циклу ReAct:*

   Thought (Думка) – знайдіть виклик до LLM (ChatGoogleGenerativeAI). Натисніть на нього. Ви побачите точний системний промпт та історію, які були надіслані моделі. В її відповіді ви побачите блок «Thought», де агент міркує, що йому робити далі.

   Action (Дія) – одразу після думки ви побачите виклик інструмента (Tool), наприклад, TavilySearch. Тут можна побачити, з якими саме параметрами агент викликав інструмент (наприклад, який пошуковий запит він сформулював).

   Observation (Спостереження) – результат, який інструмент повернув агенту. Ви можете побачити, яку саме інформацію агент отримав після виконання дії.
8. *Прослідкуйте, як агент використовує результат (Observation) для формування нової думки (Thought), і так по колу, доки не дійде до фінальної відповіді (Final Answer).*

Частина 2. Виконання індивідуального завдання (Розробка кастомного інструмента).

Обраний інструмент для створення

Назва: *Калькулятор вартості матеріалів (calculate_material_cost)*

Чому обрано: *LLM погано справляється з точними математичними розрахунками. Цей інструмент інкапсулює бізнес-логіку (ціни на матеріали) та гарантує точний результат, що є критичним у будівельній галузі.*

Опис ідеї та завдання для агента

Назва агента: *«AI-Кошторисник»*

Мета: *агент повинен автономно знаходити актуальну ринкову ціну на будівельний матеріал (використовуючи пошук), а потім розраховувати загальну вартість для заданої кількості, використовуючи кастомний інструмент калькуляції.*

Завдання для демонстрації: *«Яка буде загальна вартість 50 кубометрів бетону марки М300, якщо середня ринкова ціна на нього в Україні становить близько 3000 грн за кубометр? Збережи розрахунок у файл estimate.txt».*

1. **Створення спеціалізованого агента.**

Щоб не змінювати логіку основного чат-агента, ми створимо окремий модуль (app/agents/estimator_agent.py) для нашого кошторисника.

Визначимо інструменти для розрахунку та запису у файл, а також створимо самого агента..

```
from langchain_core.tools import tool
from pydantic import BaseModel, Field
from langchain_tavily import TavilySearch
from langgraph.prebuilt import create_react_agent
from app.config import load_google_llm

# --- Інструменти для агента ---

class MaterialCostArgs(BaseModel):
    price_per_unit: float = Field(description="Ціна за одну одиницю
матеріалу.")
    quantity: float = Field(description="Кількість одиниць матеріалу.")

@tool(args_schema=MaterialCostArgs)
def calculate_material_cost(price_per_unit: float, quantity: float) -> str:
    """Обчислює загальну вартість будівельних матеріалів."""
    total_cost = price_per_unit * quantity
    return f"Загальна вартість для {quantity} одиниць за ціною
{price_per_unit} за одиницю становить {total_cost:.2f}."
```

```
@tool
def write_report(filename: str, content: str) -> str:
    """Записує фінальний звіт у текстовий файл."""
    try:
        with open(filename, 'w', encoding='utf-8') as f:
            f.write(content)
        return f"Звіт успішно збережено у файл {filename}"
    except Exception as e:
        return f"Помилка при збереженні звіту: {e}"

# --- Функція для створення агента ---

def create_estimator_agent():
    """Створює та повертає екземпляр агента-кошторисника."""
    llm = load_google_llm()
    search_tool = TavilySearch(max_results=2)

    # Агент отримує всі необхідні інструменти
    all_tools = [search_tool, calculate_material_cost, write_report]

    agent_executor = create_react_agent(llm, tools=all_tools)
    return agent_executor
```

2. **Додавання моделей запиту та відповіді.**

Нам потрібно визначити, які дані буде приймати та повертати наш новий API (app/models/schemas.py).

```
class EstimatorRequest(BaseModel):
    task: str = Field(..., min_length=10, max_length=500,
description="Детальне завдання для кошторисника")

class EstimatorResponse(BaseModel):
    response: str
    artifact_path: str | None = None
    timestamp: datetime
```

3. **Створення нового API-ендпоінта.**

Тепер створимо «міст» між frontend та нашим новим агентом (app/routes/analysis.py).

```
from app.models.schemas import EstimatorRequest, EstimatorResponse
from app.agents.estimator_agent import create_estimator_agent

...

estimator_agent = create_estimator_agent()
```

```
@router.post("/run-estimator", response_model=EstimatorResponse)
async def run_estimator(request: EstimatorRequest):
    try:
        task_message = HumanMessage(content=request.task)
        response = await estimator_agent.ainvoke({"messages":
[task_message]})

        final_response_content = response["messages"][-1].content

        # Проста логіка для визначення, чи був створений файл
        artifact_path = "estimate.txt" if "збережено у файл" in
final_response_content.lower() else None

        return EstimatorResponse(
            response=final_response_content,
            artifact_path=artifact_path,
            timestamp=datetime.now()
        )
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Estimator agent error:
{str(e)}")
```

4. **Додавання нової вкладки та її елементів.**

Відкриємо файл frontend/index.html. Знайдемо блок . Додамо нову кнопку для вкладки:

```
<button class="tab-button" data-tab="estimator">Cost Estimator</button>
```

Після блоку ... , додамо новий блок для нашої вкладки:

```
<div id="estimator" class="tab-content">
    <input type="text" id="estimator-input" placeholder="Enter estimation
task (e.g., calculate cost for 50m³ of concrete)...">
    <button id="estimator-btn">Run Estimator</button>
</div>
```

5. **Додавання логіки для нової вкладки.**

Відкриємо файл frontend/script.js. Додамо нові константи та обробник подій для елементів кошторисника:

```
const estimatorInput = document.getElementById("estimator-input");
const estimatorBtn = document.getElementById("estimator-btn");
```

```
...

estimatorBtn.addEventListener("click", async () => {
    const task = estimatorInput.value;
    if (!task) return;

    resultsContainer.innerHTML = "Estimator is running, please wait...";
    try {
        const response = await fetch("/api/run-estimator", {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({ task }),
        });

        if (!response.ok) {
            const errorData = await response.json();
            throw new Error(errorData.detail || "Something went wrong");
        }

        const data = await response.json();
        displayEstimatorResult(data);

    } catch (error) {
        resultsContainer.innerHTML = `Error: ${error.message}`;
    }
});

function displayEstimatorResult(data) {
    let artifactHtml = "";
    if (data.artifact_path) {
        artifactHtml = `
            <h3>Artifact Created</h3>
            <p>A report file was created:
<strong>${data.artifact_path}</strong>. Check your backend project
folder.</p>
        `;
    }

    resultsContainer.innerHTML = `
        <h3>Estimator Response</h3>
        <p>${data.response}</p>
        ${artifactHtml}
    `;
}
```

6. **Запуск серверу та перевірка результату.**

1. Переконайтесь, що ваш uvicorn сервер перезапустився після змін у backend (або перезапустіть його вручну).

```
uvicorn app.main:app --reload
```

2. Оновіть сторінку в браузері (http://localhost:8000/).

3. Ви повинні побачити нову вкладку «Cost Estimator».

4. Перейдіть на неї та введіть завдання, наприклад: «Розрахуй вартість 100 квадратних метрів цегли, якщо середня ціна 450 грн за квадратний метр. Збережи результат у файл estimate.txt»

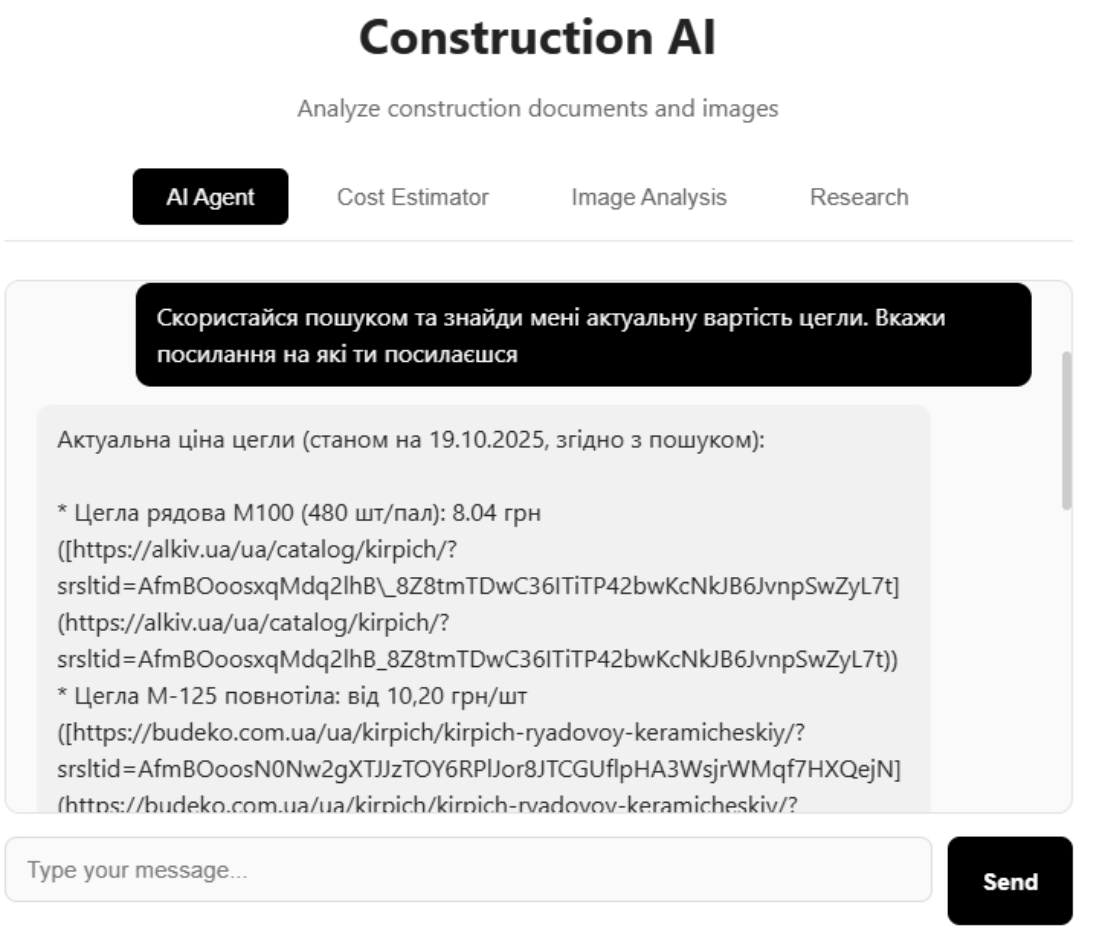
5. Натисніть «Run Estimator».

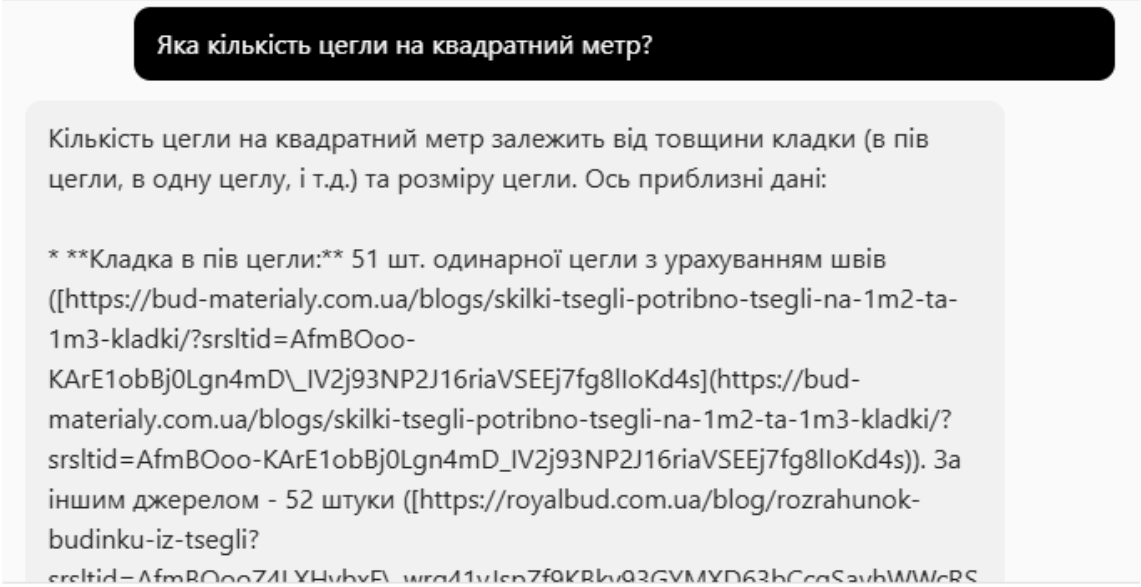
6. Очікуваний результат: в секції «Results» з'явиться відповідь від агента. В кореневій папці backend/ буде створено файл estimate.txt з розрахунком.

7. **Проаналізуй TRACE системи через smith.langchain.com.**

Після реалізації власного завдання проаналізуйте трасування для вашого нового ендпоінта. Переконайтесь, що агент коректно викликає ваш кастомний інструмент та інші інструменти згідно з вашим планом.

ПРИКЛАД РОБОТИ СИСТЕМИ





Construction AI

Analyze construction documents and images

AI Agent | Cost Estimator | Image Analysis | Research

Використай цю інформацію Цегла рядова М100 (480 шт/пал): 8.04 грн, Цегла М-125 повнотіла:

Run Estimator

Results

Estimator Response

Обчислення вартості 100 квадратних метрів кладки з використанням цегли М100: 5100 цегли x 8.04 грн = 41004 грн Обчислення вартості 100 квадратних метрів кладки з використанням цегли М125: 5100 цегли x 10.20 грн = 52020 грн Який варіант ви хочете зберегти у файл estimate.txt?

Використай цю інформацію Цегла рядова М100 (480 шт/пал): 8.04 грн, Цегла М-125 повнотіла:

Run Estimator

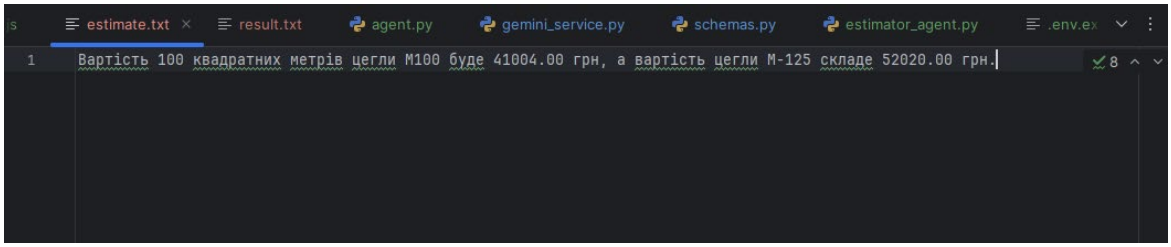
Results

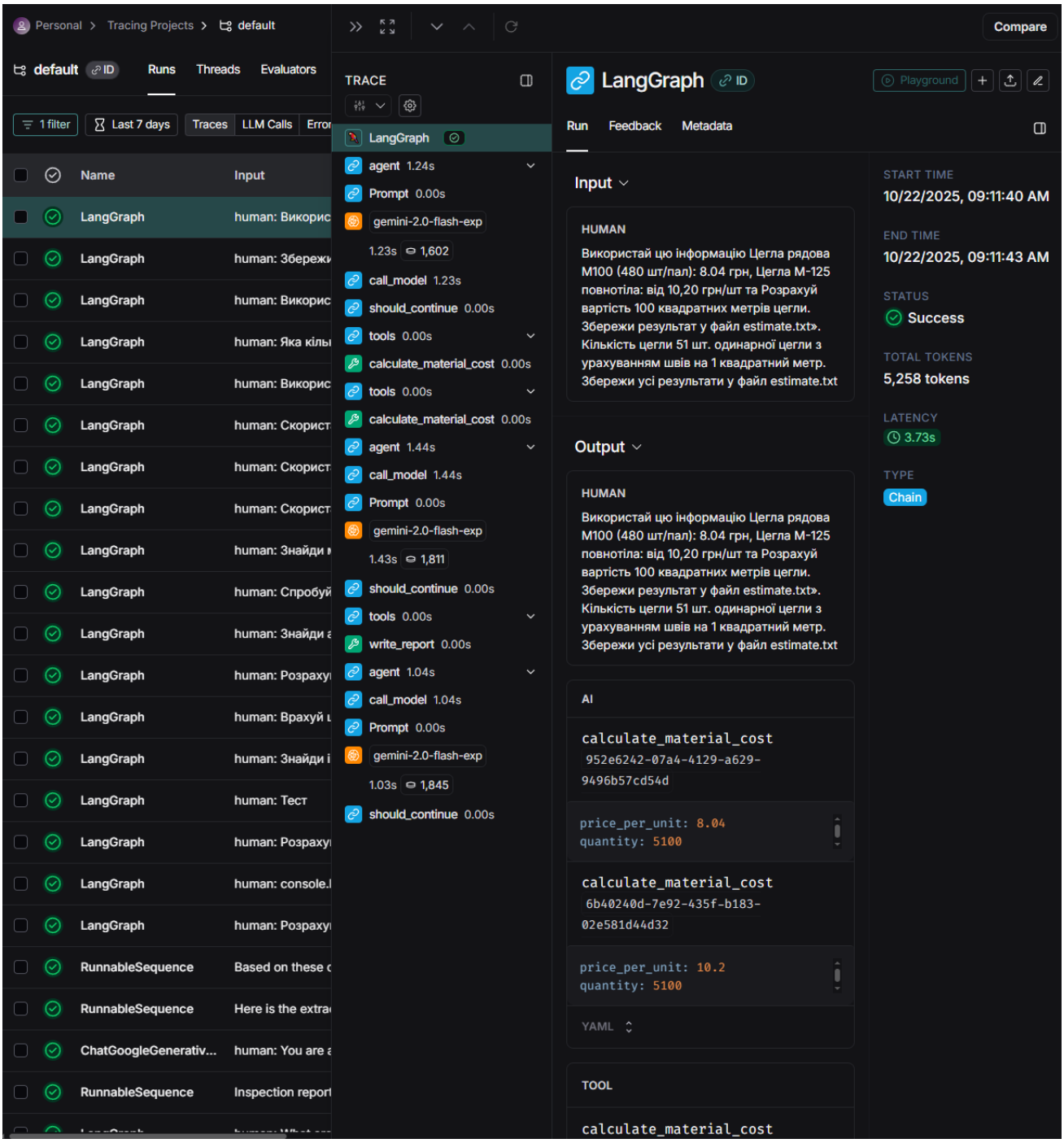
Estimator Response

Загальна вартість 100 квадратних метрів цегли М100 становить 41004.00 грн, а вартість цегли М-125 складе 52020.00 грн. Цю інформацію було збережено у файл estimate.txt.

Artifact Created

A report file was created: **estimate.txt**. Check your backend project folder.


estimate.txt
result.txt
agent.py
gemini_service.py
schemas.py
estimator_agent.py
.env.ex
1 Вартість 100 квадратних метрів цегли M100 буде 41004.00 грн, а вартість цегли M-125 складе 52020.00 грн.
8


Personal > Tracing Projects > default
default ID
Runs
Threads
Evaluators
1 filter
Last 7 days
Traces
LLM Calls
Name
Input
LangGraph human: Використ
LangGraph human: Збереж
LangGraph human: Використ
LangGraph human: Яка кіль
LangGraph human: Використ
LangGraph human: Скорист
LangGraph human: Скорист
LangGraph human: Скорист
LangGraph human: Знайди
LangGraph human: Спробуй
LangGraph human: Знайди
LangGraph human: Розрахуй
LangGraph human: Врахуй
LangGraph human: Знайди
LangGraph human: Тест
LangGraph human: Розрахуй
LangGraph human: console.l
LangGraph human: Розрахуй
RunnableSequence Based on these
RunnableSequence Here is the extra
ChatGoogleGenerativ... human: You are
RunnableSequence Inspection report
TRACE
LangGraph
agent 1.24s
Prompt 0.00s
gemini-2.0-flash-exp
1.23s 1,602
call_model 1.23s
should_continue 0.00s
tools 0.00s
calculate_material_cost 0.00s
tools 0.00s
calculate_material_cost 0.00s
agent 1.44s
call_model 1.44s
Prompt 0.00s
gemini-2.0-flash-exp
1.43s 1,811
should_continue 0.00s
tools 0.00s
write_report 0.00s
agent 1.04s
call_model 1.04s
Prompt 0.00s
gemini-2.0-flash-exp
1.03s 1,845
should_continue 0.00s
Compare
LangGraph ID
Playground
Run
Feedback
Metadata
Input
HUMAN
Використай цю інформацію Цегла рядова M100 (480 шт/пал): 8.04 грн, Цегла M-125 повнотіла: від 10,20 грн/шт та Розрахуй вартість 100 квадратних метрів цегли. Збережи результат у файл estimate.txt». Кількість цегли 51 шт. одинарної цегли з урахуванням швів на 1 квадратний метр. Збережи усі результати у файл estimate.txt
Output
HUMAN
Використай цю інформацію Цегла рядова M100 (480 шт/пал): 8.04 грн, Цегла M-125 повнотіла: від 10,20 грн/шт та Розрахуй вартість 100 квадратних метрів цегли. Збережи результат у файл estimate.txt». Кількість цегли 51 шт. одинарної цегли з урахуванням швів на 1 квадратний метр. Збережи усі результати у файл estimate.txt
AI
calculate_material_cost
952e6242-07a4-4129-a629-9496b57cd54d
price_per_unit: 8.04
quantity: 5100
calculate_material_cost
6b40240d-7e92-435f-b183-02e581d44d32
price_per_unit: 10.2
quantity: 5100
YAML
TOOL
calculate_material_cost
START TIME
10/22/2025, 09:11:40 AM
END TIME
10/22/2025, 09:11:43 AM
STATUS
Success
TOTAL TOKENS
5,258 tokens
LATENCY
3.73s
TYPE
Chain

ПРИКЛАДИ ІНДИВІДУАЛЬНИХ ЗАВДАНЬ (НЕОБОВ'ЯЗКОВО)

Нижче наведено кілька ідей для розробки власних AI-агентів. Ви можете обрати одну з них або, що заохочується, розробити власну концепцію на основі цих прикладів. Головна мета – продемонструвати здатність агента автономно вирішувати задачу, комбінуючи різні інструменти.

Варіант 1. AI-інженер-консультант з конвертації одиниць

Назва агента: «AI-Конвертер»

Опис ідеї: у будівництві часто доводиться працювати з різними системами вимірювань (метричною та імперською), а також конвертувати складні одиниці (наприклад, щільність, тиск). LLM може робити помилки в точних розрахунках. Завдання – створити надійний інструмент для конвертації та навчити агента ним користуватися.

Приклад завдання для агента: «Мені потрібно замовити 2500 футів арматури для об'єкта в Європі. Переведи цю довжину в метри. Також, знайди в інтернеті, яка стандартна вага одного метра арматури діаметром 12 мм. Збережи обидва результати у файл conversion_report.txt».

Технічні поради:

1. *Створіть інструмент convert_units:*

Ваша функція може приймати параметри: value: float, from_unit: str, to_unit: str.

Всередині функції використовуйте словник з коефіцієнтами перерахунку (напр., {'foot_to_meter': 0.3048}).

Інструмент має повертати рядок з результатом.

2. *Інтеграція в API: створіть новий ендпоінт, наприклад, /api/convert-units.*
3. *Frontend: додайте вкладку «Unit Converter», поле для введення текстового завдання та кнопку запуску. Результат (відповідь агента та інформація про створений файл) виводьте в блок «Results».*

Варіант 2. AI-Інспектор з безпеки

Назва агента: «AI-безпека»

Опис ідеї: перед виконанням будь-яких будівельних робіт необхідно складати чек-лист з техніки безпеки. Агент може автоматизувати цей процес, використовуючи як вбудовану базу знань, так і актуальну інформацію з інтернету.

Приклад завдання для агента: «Підготуй чек-лист з техніки безпеки для виконання зварювальних робіт на висоті. Використай свою базу знань для основних пунктів, а також знайди в інтернеті 2-3 додаткові сучасні рекомендації. Сформуй фінальний чек-лист та збережи його у файл safety_checklist_welding.md».

Технічні поради:

1. *Створіть інструмент get_base_safety_checklist:*

Функція може приймати один параметр: work_type: str (напр., "welding", "high-altitude work").

Всередині функції створіть словник, де ключем є тип робіт, а значенням – список базових правил безпеки.

Інструмент має повертати рядок з відформатованим списком.

2. *Комбінація інструментів: агент повинен буде спочатку викликати ваш кастомний інструмент, потім – TavilySearch для пошуку додаткової інформації, а наприкінці – write_report для збереження об'єднаного результату.*
3. *Інтеграція в API: створіть ендпоінт /api/generate-checklist.*
4. *Frontend: додайте вкладку «Safety Inspector» з полем для опису типу робіт.*

Варіант 3. AI-Планувальник Проєктів

Назва агента: «AI-Планувальник»

Опис ідеї: агент може допомогти у створенні чорнового плану-графіку для невеликих будівельних проєктів, розбиваючи велику задачу на етапи та визначаючи їх послідовність.

Приклад завдання для агента: «Створи простий план-графік для будівництва фундаменту. Він має включати наступні етапи: 1. Риття котловану, 2. Встановлення опалубки, 3. В'язка арматурного каркасу, 4. Заливка бетону. Оціни приблизну тривалість кожного етапу в днях і збережи план у вигляді Markdown-таблиці у файл foundation_plan.md».

Технічні поради:

1. *Створіть інструмент add_plan_step_to_file:*

Цей інструмент буде більш складним. Він може приймати filename: str, step_name: str, duration_days: int, dependencies: str.

Функція має дописувати новий рядок у вказаний файл, форматуючи його як рядок Markdown-таблиці (| Назва етапу | Тривалість | Залежності |).

2. *Складний воркфлоу: завдання вимагатиме від агента зробити кілька послідовних викликів одного й того ж інструмента для кожного етапу плану.*
3. *Інтеграція в API: створіть ендпоінт /api/create-project-plan.*
4. *Frontend: додайте вкладку «Project Planner» з полем для загального опису проєкту.*

Ви можете придумати власні інструменти: для пошуку постачальників, для аналізу будівельних кодів, для генерації щоденних звітів тощо.

ІНСТРУКЦІЇ З НАЛАШТУВАННЯ ТА ЗАПУСКУ

Отримання API-ключів

Перед початком роботи переконайтесь, що у вас є безкоштовні API-ключі від наступних сервісів:

Google Gemini API key: тиць

Tavily API key: тиць

LangSmith API key (опційно, але рекомендовано): тиць

Клонування та налаштування проєкту

Відкрийте термінал або командний рядок та виконайте наступні команди:

```
git clone https://github.com/F1-bot/langchain_sample.git
cd construction-ai/backend

python -m venv venv

Windows: venv\Scripts\activate
Mac/Linux: source venv/bin/activate

pip install -r requirements.txt
```

Конфігурація середовища

Створіть файл *.env* у папці *backend/* (можна скопіювати з *.env.example*).

Відкрийте *.env* та вставте ваші API-ключі, отримані на першому кроці.

Запуск сервера

```
uvicorn app.main:app --reload
```

Тестування та виконання роботи

http://127.0.0.1:8000/ & http://127.0.0.1:8000/docs

https://smith.langchain.com/

Welcome to Github Repository

github.com/F1-bot/langchain_sample.