Створення Безпечної Мікро Інформаційно-Аналітичної Системи: Практичний Посібник для Аналітиків (VS Code Edition)

Частина I: Фундамент та Налаштування Середовища

Ця частина є найважливішою для початківця. Невдача на цьому етапі означає, що проєкт ніколи не розпочнеться. Мета полягає в тому, щоб зміцнити впевненість шляхом методичного подолання початкових, і часто найбільш неприємних, технічних перешкод, пов'язаних із встановленням та конфігурацією програмного забезпечення на Windows.

Розділ 1: Підготовка вашого цифрового робочого простору на Windows

1.1 Вступ: Три стовпи нашої системи

Перш ніж розпочати, важливо зрозуміти архітектуру системи, яку ми створюємо. Її можна уявити як структуру, що складається з трьох основних компонентів, або "стовпів", кожен з яких виконує свою унікальну роль:

1. **База даних PostgreSQL (Пам'ять):** Це надійне сховище для всієї вашої інформації. Уявіть її як високоорганізовану цифрову картотеку, де кожен документ має своє чітко визначене місце. PostgreSQL відповідає за безпечне зберігання, структурування та

швидкий доступ до даних.
2. **Сервер Flask (Мозок):** Це логічний центр системи. Flask, написаний на мові програмування Python, діє як посередник. Він отримує запити від користувацького інтерфейсу (наприклад, "показати всі документи"), обробляє їх, взаємодіє з базою даних, щоб отримати або зберегти інформацію, і надсилає результат назад. Це "мозок", що керує всіма операціями.
3. **Веб-інтерфейс (Обличчя):** Це те, що бачить і з чим взаємодіє аналітик у своєму веб-браузері. Це візуальне представлення даних у вигляді таблиць, кнопок та полів для введення. Створений за допомогою стандартних веб-технологій (HTML, CSS, JavaScript), цей компонент дозволяє зручно переглядати, додавати та видаляти документи.

Розуміння цієї трикомпонентної моделі допоможе вам краще орієнтуватися в процесі розробки, оскільки кожен наступний крок буде спрямований на створення або налаштування одного з цих стовпів.

1.2 Встановлення та конфігурація Python 3.11+

Python — це мова програмування, яка стане основою для логіки нашого сервера. Правильне встановлення є першим і найважливішим кроком.

1. **Завантаження інсталятора:** Перейдіть на офіційний сайт Python за адресою python.org. На головній сторінці ви побачите кнопку для завантаження останньої стабільної версії для Windows. Рекомендується використовувати версію 3.11 або новішу.[1]
2. **Процес встановлення:** Запустіть завантажений файл .exe. Перед вами з'явиться вікно інсталятора. Тут є один критично важливий крок, який часто пропускають новачки.
   ВАЖЛИВО: У першому вікні інсталяції обов'язково встановіть прапорець біля опції **"Add Python to PATH"** ("Додати Python до PATH").[2]
   Що таке PATH? Це системна змінна середовища у Windows, яка містить список директорій. Коли ви вводите команду (наприклад, python) у командний рядок, Windows шукає виконуваний файл цієї команди у всіх директоріях, перелічених у PATH. Якщо не встановити цей прапорець, Windows "не знатиме", де знайти Python, і вам доведеться вказувати повний шлях до нього щоразу, що дуже незручно і може викликати помилки в роботі інших інструментів.
   Після встановлення прапорця натисніть **"Install Now"** ("Встановити зараз"), щоб розпочати стандартну інсталяцію.
3. **Перевірка встановлення:** Щоб переконатися, що Python встановлено правильно, відкрийте командний рядок. Для цього натисніть клавішу Win, введіть PowerShell і

запустіть знайдену програму. У вікні, що відкрилося, введіть по черзі наступні команди, натискаючи Enter після кожної:
PowerShell
python --version
pip --version

Якщо встановлення пройшло успішно, перша команда виведе версію Python (наприклад, Python 3.11.5), а друга — версію pip, менеджера пакунків Python, який встановлюється разом з ним. Якщо ви бачите ці повідомлення, ви готові до наступного кроку. Якщо з'являється помилка, що команда не розпізнана, найімовірніше, ви пропустили крок із додаванням Python до PATH. У такому разі найкраще видалити Python через "Панель керування" та встановити його знову, приділивши увагу цьому прапорцю.

1.3 Встановлення PostgreSQL

PostgreSQL (часто називають просто Postgres) — це потужна та надійна система управління базами даних з відкритим вихідним кодом. Ми будемо використовувати її для зберігання даних про документи.

1. **Завантаження інсталятора:** Найпростіший спосіб встановити PostgreSQL на Windows — це використати інсталятор від компанії EDB. Перейдіть на офіційну сторінку завантажень PostgreSQL (postgresql.org/download/windows/) і виберіть останню стабільну версію для завантаження.[4]
2. **Процес встановлення:** Запустіть завантажений інсталятор від імені адміністратора (правою кнопкою миші -> "Запустити з правами адміністратора").[6] Дотримуйтесь інструкцій майстра встановлення:
    - **Installation Directory (Директорія встановлення):** Залиште шлях за замовчуванням (C:\Program Files\PostgreSQL\<version>).[7]
    - **Select Components (Вибір компонентів):** Переконайтеся, що обрані щонайменше PostgreSQL Server та Command Line Tools ("Інструменти командного рядка"). Графічний клієнт pgAdmin встановлювати не обов'язково, оскільки ми будемо працювати з базою даних безпосередньо з VS Code.[7]
    - **Data Directory (Директорія даних):** Можна залишити шлях за замовчуванням.[7]
    - **Password (Пароль):** На цьому етапі вам потрібно встановити пароль для головного користувача (суперкористувача) бази даних, який називається postgres. **Це надзвичайно важливий крок.** Виберіть надійний пароль, запишіть його в безпечному місці (менеджері паролів) і запам'ятайте. Цей пароль буде потрібен для доступу до бази даних.[6]
    - **Port (Порт):** Залиште порт за замовчуванням — 5432.[6]

- **Locale (Локаль):** Залиште налаштування за замовчуванням (Default locale).[7]
- Завершіть встановлення.

1.4 Встановлення Visual Studio Code та ключових розширень

Visual Studio Code (VS Code) — це легкий, але потужний редактор коду, який стане нашим єдиним центром для розробки.

1. **Завантаження та встановлення:** Перейдіть на офіційний сайт code.visualstudio.com та завантажте інсталятор для Windows. Процес встановлення є стандартним.
2. **Встановлення розширень:** Розширення перетворюють VS Code на потужний інструмент для конкретних завдань. Ми встановимо три основні розширення.
   - Запустіть VS Code.
   - На панелі зліва знайдіть іконку з квадратами (Extensions) або натисніть Ctrl+Shift+X.
   - У рядку пошуку по черзі знайдіть та встановіть наступні розширення:
     1. **Python** (від Microsoft): Забезпечує підсвічування синтаксису, автодоповнення коду, налагодження та інші функції для розробки на Python.[3]
     2. **PostgreSQL** (від Microsoft): Дозволяє підключатися до баз даних PostgreSQL, переглядати їх структуру, виконувати SQL-запити та бачити результати безпосередньо у VS Code, замінюючи зовнішні інструменти, такі як pgAdmin або psql.[26]
     3. **Live Server** (від Ritwick Dey): Запускає локальний сервер для розробки, який автоматично оновлює веб-сторінку у браузері при збереженні змін у файлах HTML, CSS або JavaScript. Це значно прискорює розробку фронтенду.[29]

Після виконання цих кроків ваш комп'ютер повністю готовий до розробки в єдиному, інтегрованому середовищі.

Розділ 2: Інструментарій аналітика в середовищі VS Code

2.1 Робота у вбудованому терміналі VS Code

Замість використання окремого вікна PowerShell, ми будемо працювати у **вбудованому**

терміналі (Integrated Terminal) VS Code. Це дозволяє виконувати всі команди, не виходячи з редактора коду.

- **Як відкрити термінал:** Натисніть комбінацію клавіш Ctrl+` (зворотний апостроф, зазвичай знаходиться на клавіші з літерою Ё або ~). Внизу вікна VS Code з'явиться панель терміналу.[32]
- **Переваги:** Вбудований термінал автоматично відкривається в кореневій папці вашого проєкту. Розширення Python також інтегрується з ним, допомагаючи автоматично активувати правильне віртуальне середовище.[33]

Всі подальші команди, якщо не вказано інше, слід виконувати саме у вбудованому терміналі VS Code.

2.2 Управління проєктом за допомогою віртуальних середовищ (venv)

Віртуальне середовище (virtual environment) створює ізольовану "майстерню" для вашого проєкту, щоб уникнути конфліктів бібліотек.[10] Модуль

venv, що входить до складу Python, дозволяє створювати такі середовища.[11]

1. **Створення віртуального середовища:**
   - Відкрийте VS Code. Використовуйте меню File > Open Folder... і виберіть вашу папку проєкту (напр., micro_ias).
   - Відкрийте вбудований термінал (`Ctrl+``).
   - Виконайте команду для створення віртуального середовища з назвою .venv:
     PowerShell
     python -m venv.venv

     Ця команда створить у вашій проєктній директорії папку .venv.[12]
2. **Активація віртуального середовища:**
   - **Критичний крок для Windows:** За замовчуванням PowerShell може блокувати виконання локальних скриптів. Щоб це виправити, потрібно один раз змінити політику виконання. Виконайте цю команду у вбудованому терміналі:
     PowerShell
     Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

     На запит підтвердження введіть Y і натисніть Enter.[11]
   - Тепер активуйте середовище, виконавши таку команду:

     ..venv\Scripts\activate```
   - Після успішної активації на початку вашого командного рядка з'явиться (.venv).[14]

VS Code також може автоматично виявити це середовище і запропонувати використовувати його для проєкту.

3. **Деактивація:** Щоб вийти з віртуального середовища, просто введіть команду deactivate.

2.3 Встановлення пакунків за допомогою pip

pip — це система управління пакунками для Python. Для відстеження залежностей проєкту використовується файл requirements.txt.

- **Встановлення з файлу:**
  PowerShell
  `pip install -r requirements.txt`

- **Створення файлу:**
  PowerShell
  `pip freeze > requirements.txt`

2.4 Шпаргалка команд (Cheat Sheet)

Команда (Command)	Опис (Українська)	Опис (English)	Контекст / Коли використовувати
Ctrl+Shift+P	Відкрити Палітру Команд VS Code.	Open the VS Code Command Palette.	Для доступу до будь-якої команди VS Code або розширень.
Ctrl+`	Відкрити/закрити вбудований термінал.	Toggle the integrated terminal.	Для виконання команд python, pip, flask.
python -m venv.venv	Створити віртуальне	Create a virtual environment named	На самому початку проєкту, один раз.

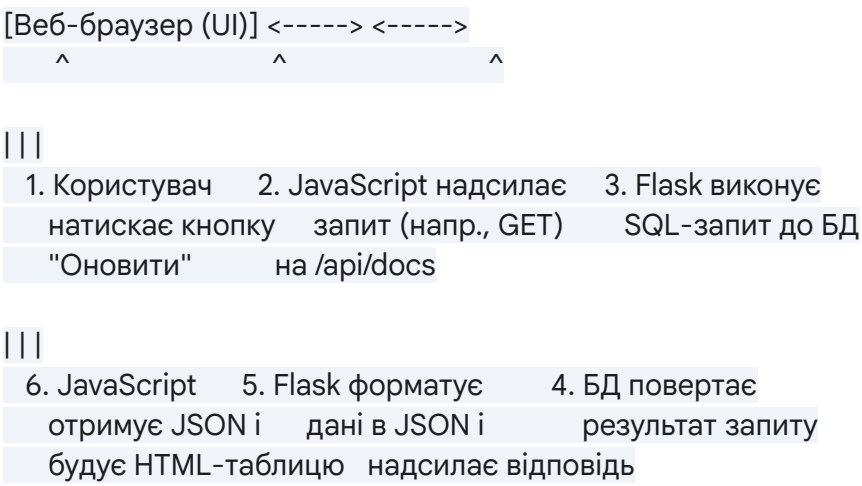
	середовище .venv.	.venv.	
.\.venv\Scripts\activate	Активувати віртуальне середовище.	Activate the virtual environment.	Кожного разу, коли ви починаєте працювати над проєктом у новому терміналі.
pip install -r requirements.txt	Встановити залежності з файлу.	Install dependencies from the file.	Після клонування проєкту або для налаштування середовища.
pip freeze > requirements.txt	Зберегти залежності у файл.	Save dependencies to the file.	Після встановлення нового пакунка.
flask run	Запустити локальний сервер Flask.	Start the local Flask development server.	Для тестування вашого веб-додатку.
Right-click > Open with Live Server	(На файлі index.html) Запустити фронтенд.	(On index.html file) Launch the frontend.	Для перегляду та тестування вашого UI.

Розділ 3: Огляд архітектури системи

3.1 Погляд з висоти пташиного польоту

Діаграма ілюструє потік інформації від користувача до бази даних і назад.

Діаграма потоку даних:



Покроковий сценарій (отримання списку документів):

1. **Дія користувача:** Аналітик відкриває веб-сторінку у своєму браузері.
2. **Запит від клієнта:** Код JavaScript надсилає мережевий запит на сервер Flask, наприклад, GET /api/docs.
3. **Обробка на сервері:** Сервер Flask отримує запит і підключається до бази даних PostgreSQL.
4. **Запит до бази даних:** Flask формує SQL-запит (наприклад, SELECT * FROM inbound_docs;) і надсилає його до PostgreSQL.
5. **Відповідь від бази даних:** PostgreSQL виконує запит і повертає дані серверу Flask.
6. **Форматування відповіді:** Flask перетворює дані у формат **JSON** і надсилає його як відповідь браузеру.
7. **Відображення даних:** JavaScript у браузері отримує JSON, динамічно створює рядки для HTML-таблиці та вставляє їх на сторінку.

Частина II: Рівень даних - Пам'ять системи

Ця частина присвячена базі даних. Ми будемо використовувати розширення **PostgreSQL** у VS Code для всіх операцій, що робить процес візуальним та інтегрованим.

Розділ 4: Проєктування таблиці inbound_docs

4.1 Що таке схема бази даних?

Схема бази даних (database schema) — це її креслення. Вона визначає, які таблиці існують, які стовпці є в кожній таблиці, і якого типу дані можуть зберігатися в цих стовпцях.

4.2 Таблиця inbound_docs: обґрунтування полів

Для нашої системи ми створимо одну таблицю inbound_docs. Кожне поле (стовпець) ретельно підібране для задоволення мінімальних потреб військового аналітика.[15]

- **id (SERIAL PRIMARY KEY):** Унікальний ідентифікатор запису, що генерується автоматично.
- **doc_reference (VARCHAR(255) UNIQUE NOT NULL):** Людино-зрозумілий номер документа (напр., "INTREP-2024-001").
- **received_at (TIMESTAMPTZ NOT NULL DEFAULT NOW()):** Точний час і дата реєстрації документа в системі, з урахуванням часового поясу.
- **source_type (VARCHAR(50)):** Тип джерела розвідданих ('HUMINT', 'OSINT', 'SIGINT' тощо).[17]
- **classification (VARCHAR(50) NOT NULL):** Рівень таємності документа ('UNCLASSIFIED', 'SECRET' тощо).
- **summary (TEXT NOT NULL):** Короткий виклад суті документа.
- **analyst_notes (TEXT):** Довільні нотатки аналітика.
- **priority (INTEGER DEFAULT 3):** Пріоритет документа (1-Високий, 2-Середній, 3-Низький).

4.3 Повний SQL-скрипт

Збережіть цей код у файл з назвою schema.sql.

SQL

```sql
-- Видаляємо таблицю, якщо вона вже існує, щоб уникнути помилок при повторному запуску скрипта
DROP TABLE IF EXISTS inbound_docs;

-- Створення таблиці для обліку вхідних документів
CREATE TABLE inbound_docs (
    id SERIAL PRIMARY KEY,
    doc_reference VARCHAR(255) UNIQUE NOT NULL,
    received_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    source_type VARCHAR(50),
    classification VARCHAR(50) NOT NULL,
    summary TEXT NOT NULL,
    analyst_notes TEXT,
    priority INTEGER DEFAULT 3
);

-- Створення індексів для прискорення пошуку
CREATE INDEX idx_received_at ON inbound_docs(received_at);
CREATE INDEX idx_source_type ON inbound_docs(source_type);

COMMENT ON TABLE inbound_docs IS 'Таблиця для зберігання метаданих про вхідні аналітичні документи.';
```

Розділ 5: Наповнення та взаємодія з базою даних у VS Code

Тепер, коли у нас є "креслення" (схема), час "побудувати" нашу базу даних і заселити її даними, використовуючи графічний інтерфейс VS Code.

5.1 Підключення до сервера PostgreSQL

1. У VS Code на панелі зліва натисніть на іконку розширення **PostgreSQL** (слон).
2. У верхній частині панелі, що відкрилася, натисніть кнопку **"+" (Add Connection)**.[26]
3. VS Code послідовно запитає у вас параметри підключення у верхній частині екрана:
    - **Hostname:** Введіть localhost і натисніть Enter.[35]

- **Username:** Введіть postgres і натисніть Enter.
- **Password:** Введіть пароль, який ви встановили для користувача postgres.
- **Port:** Залиште 5432 і натисніть Enter.
- **Connection:** Виберіть Standard Connection.
- **Show all databases:** Виберіть цей варіант.
- **Display Name:** Можете залишити за замовчуванням (localhost) або дати будь-яке ім'я.

4. Якщо все правильно, у панелі PostgreSQL з'явиться ваше підключення до сервера localhost.[35]

5.2 Створення бази даних та застосування схеми

1. **Створення бази даних:**
   - У панелі PostgreSQL клацніть правою кнопкою миші на вашому сервері (localhost) і виберіть **New Query**.[26]
   - Відкриється новий редактор SQL. Введіть у нього команду:
     ```SQL
     CREATE DATABASE military_docs;
     ```
   - Виділіть цей рядок, клацніть правою кнопкою миші та виберіть **Run Query**.[35] У нижній панелі з'явиться повідомлення про успішне виконання.
   - Клацніть правою кнопкою миші на сервері та виберіть **Refresh**, щоб побачити нову базу даних military_docs у списку.
2. **Застосування схеми:**
   - Тепер клацніть правою кнопкою миші на щойно створеній базі даних **military_docs** і виберіть **New Query**. Це гарантує, що наступні команди будуть виконуватися саме в цій базі.
   - Відкрийте ваш файл schema.sql в редакторі VS Code.
   - Скопіюйте весь його вміст у вікно "New Query".
   - Натисніть правою кнопкою миші у вікні запиту та виберіть **Run Query**. Таблиця inbound_docs та індекси будуть створені. Ви можете розгорнути вашу базу даних у бічній панелі, щоб побачити нову таблицю.[28]

5.3 Імпорт зразкового набору даних

Створіть файл sample_data.sql і скопіюйте в нього вміст з Додатку С.

1. Переконайтеся, що у вас відкрите вікно запиту, підключене до бази military_docs.
2. Відкрийте файл sample_data.sql, скопіюйте його вміст, вставте у вікно запиту і виконайте його (**Run Query**). 15 записів будуть додані до вашої таблиці.

5.4 Перші запити: розмова з вашою базою даних

Тепер ви можете легко взаємодіяти з даними.

- Клацніть правою кнопкою миші на таблиці inbound_docs у бічній панелі та виберіть **Select Top 1000**. VS Code автоматично виконає запит SELECT * FROM "public"."inbound_docs" LIMIT 1000; і покаже результати у зручній таблиці в новому вікні.[26]
- Використовуйте вікно **New Query** для виконання будь-яких інших SQL-запитів. Розширення надає підсвічування синтаксису та автодоповнення, що робить написання запитів набагато простішим.[28]

Частина III: Рівень логіки - Мозок системи

Ця частина присвячена створенню "мозку" нашої системи — бекенду на Flask.

Розділ 6: Структурування бекенду на Flask

6.1 Патерн "Фабрика додатків" для початківців

Ми розділимо код на логічні частини для кращої організації.[18]

Структура проєкту:

```
micro_ias/
├──.venv/
├── app/
│   ├── __init__.py
│   ├── db.py
│   └── routes.py
├── frontend/
│   ├── index.html
│   ├── style.css
│   └── app.js
├──.env
├── requirements.txt
└── run.py
```

Розділ 7: Створення API для документів

Спочатку встановимо необхідні бібліотеки. Створіть файл requirements.txt та додайте до нього рядки:

```
Flask
psycopg2-binary
python-dotenv
Flask-Cors
```

Тепер у вбудованому терміналі VS Code (з активованим .venv) виконайте:

PowerShell

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

7.1 Налаштування з'єднання з базою даних (db.py)

1. Створіть файл .env у кореневій папці проєкту. Додайте до нього рядок, замінивши <your_password> на ваш пароль:
   DATABASE_URL=postgresql://postgres:<your_password>@localhost:5432/military_docs

2. Напишіть код у файлі app/db.py:
   Python
   ```
   # app/db.py
   import os
   import psycopg2
   from dotenv import load_dotenv

   load_dotenv()

   def get_db_connection():
       """Встановлює з'єднання з базою даних."""
       try:
           conn = psycopg2.connect(os.getenv('DATABASE_URL'))
           return conn
       except psycopg2.OperationalError as e:
           print(f"Помилка підключення до бази даних: {e}")
           return None
   ```

7.2 Реалізація API ендпоінтів (routes.py)

Створіть логіку для ендпоінтів у файлі app/routes.py. Повний код наведено в Додатку В.

7.3 Збірка додатку (__init__.py та run.py)

Об'єднайте все разом. Повні коди файлів app/__init__.py та run.py наведені в Додатку В.

Щоб запустити ваш бекенд, відкрийте вбудований термінал VS Code (`Ctrl+``), переконайтеся, що віртуальне середовище активоване, і виконайте команду:

PowerShell

```
flask run
```

VS Code, завдяки розширенню Python, допоможе вам переконатися, що ви використовуєте правильний інтерпретатор з вашого .venv середовища.[38] Сервер буде запущено на

http://127.0.0.1:5000.

Розділ 8: Зміцнення вашого бекенду: основні практики безпеки

8.1 Запобігання SQL-ін'єкціям: золоте правило

Ніколи не форматуйте SQL-запити, вставляючи в них дані від користувача напряму. Завжди використовуйте **параметризовані запити**, як це зроблено в нашому коді routes.py. Драйвер psycopg2 бере на себе відповідальність за безпечну вставку даних.[21]

Python

```
# БЕЗПЕЧНО І ПРАВИЛЬНО
sql = "INSERT INTO inbound_docs (summary) VALUES (%s);"
data_tuple = (user_summary,)
cursor.execute(sql, data_tuple)
```

8.2 Налаштування Cross-Origin Resource Sharing (CORS)

CORS — це механізм безпеки браузера. Оскільки наш фронтенд (запущений Live Server) і бекенд (Flask) працюють на різних портах, нам потрібно дозволити їм взаємодіяти. Розширення Flask-CORS робить це за нас.[24]

8.3 Базова валідація на стороні сервера

Ніколи не довіряйте даним, що надходять від клієнта. Завжди перевіряйте їх на стороні сервера, як це зроблено в ендпоінті add_doc.

Частина IV: Рівень представлення - Обличчя системи

Розділ 9: Створення користувацького інтерфейсу за допомогою HTML та CSS

Створіть файли frontend/index.html та frontend/style.css. Їхній повний код наведено в Додатку В.

Розділ 10: Оживлення даних за допомогою JavaScript

Створіть файл frontend/app.js. Його повний код наведено в Додатку В.

Розділ 11: Принципи дружнього до аналітика інтерфейсу

Наш інтерфейс дотримується ключових принципів UX: **чіткість**, **зворотний зв'язок** та

ефективність.

Частина V: Збірка та усунення несправностей

Розділ 12: Повна збірка: чек-лист виконання

1. **[ ] Крок 1: Встановити Python 3.11+** (з прапорцем "Add Python to PATH").
2. **[ ] Крок 2: Встановити PostgreSQL**.
3. **[ ] Крок 3: Встановити VS Code** та розширення: **Python**, **PostgreSQL** (Microsoft), **Live Server** (Ritwick Dey).
4. **[ ] Крок 4: Створити папку проєкту** та відкрити її у VS Code (File > Open Folder...).
5. **[ ] Крок 5: Створити та активувати віртуальне середовище** у вбудованому терміналі VS Code.
6. **[ ] Крок 6: Створити структуру файлів та папок** згідно з розділом 6.1.
7. **[ ] Крок 7: Створити requirements.txt** та встановити залежності (pip install -r requirements.txt).
8. **[ ] Крок 8: Підключитися до PostgreSQL** через розширення VS Code та створити базу даних military_docs за допомогою New Query.
9. **[ ] Крок 9: Створити schema.sql та sample_data.sql**. Виконати їх у VS Code через Run Query для бази military_docs.
10. **[ ] Крок 10: Створити файл .env** з рядком підключення до бази даних.
11. **[ ] Крок 11: Наповнити кодом файли бекенду**: app/db.py, app/routes.py, app/__init__.py, run.py.
12. **[ ] Крок 12: Запустити бекенд-сервер** у вбудованому терміналі VS Code командою flask run.
13. **[ ] Крок 13: Наповнити кодом файли фронтенду**: frontend/index.html, frontend/style.css, frontend/app.js.
14. **[ ] Крок 14: Запустити фронтенд**: у файловому провіднику VS Code клацнути правою кнопкою миші на frontend/index.html і вибрати **Open with Live Server**.[29]
15. **[ ] Крок 15: Перевірити функціональність**: у браузері, що відкрився, таблиця має завантажити дані; кнопки, сортування та фільтрація мають працювати.

Розділ 13: Поширені помилки та швидкі рішення (FAQ)

Симптом / Повідомлення про помилку	Ймовірна причина	Рішення (Дія у VS Code)
The term 'python' is not recognized... (у терміналі)	Python не був доданий до системної змінної PATH.	Перевстановити Python з прапорцем "Add Python to PATH".
Помилка підключення до БД у розширенні PostgreSQL.	Неправильний хост, порт, ім'я користувача або пароль.	Перевірте параметри підключення. Переконайтеся, що сервер PostgreSQL запущено.
ModuleNotFoundError: No module named 'flask'	1. Віртуальне середовище не активоване. 2. Пакунки не встановлені.	1. Активуйте середовище: .\.venv\Scripts\activate. 2. Встановіть залежності: pip install -r requirements.txt.
У браузері порожня таблиця, а в консолі розробника (F12) помилка CORS.	1. Сервер Flask не запущено. 2. У коді Flask не налаштовано Flask-CORS.	1. Переконайтеся, що в терміналі VS Code запущено flask run. 2. Перевірте наявність CORS(app) у файлі app/__init__.py.
Кнопка "Go Live" або опція "Open with Live Server" відсутня.	1. Розширення Live Server не встановлено/вимкнено. 2. Не відкрито папку (workspace), а лише окремий файл.	1. Перевірте, чи встановлено та активовано розширення. 2. Відкрийте всю папку проєкту через File > Open Folder....[31]
500 Internal Server Error при додаванні/видаленні.	Помилка в Python-коді на бекенді.	Уважно прочитайте повний текст помилки у вбудованому терміналі VS Code, де запущено Flask. Він вкаже на проблемний

		файл і рядок.

Додатки

Додаток А: Глосарій термінів (Українська ⟶ English)

Термін (Українська)	Термін (English)	Опис
API (Інтерфейс прикладного програмування)	API (Application Programming Interface)	Набір правил та інструментів, що дозволяє різним програмам "спілкуватися" між собою.
CRUD	CRUD (Create, Read, Update, Delete)	Акронім для чотирьох базових операцій з даними: Створення, Читання, Оновлення, Видалення.
Бекенд	Backend	"Серверна" частина додатку, яка працює на сервері та відповідає за логіку, обробку даних та взаємодію з базою даних.
Віртуальне середовище	Virtual Environment	Ізольований простір для проєкту Python, що дозволяє уникнути конфліктів залежностей між різними проєктами.
Ендпоінт (Кінцева точка)	Endpoint	Конкретна URL-адреса на API, до якої можна

		надсилати запити для виконання певної операції (напр., /api/docs).
Змінна середовища	Environment Variable	Конфігураційна змінна, що зберігається поза кодом додатку, на рівні операційної системи.
CORS (Спільне використання ресурсів між різними джерелами)	CORS (Cross-Origin Resource Sharing)	Механізм безпеки браузера, який контролює, чи може веб-сторінка з одного домену робити запити до ресурсів на іншому домені.
Пагінація	Pagination	Розбиття великого набору даних на окремі сторінки для зручного відображення.
SQL-ін'єкція	SQL Injection	Тип атаки, при якій зловмисник вставляє шкідливий SQL-код у запит до бази даних.
Схема бази даних	Database Schema	Формальний опис структури бази даних: таблиць, стовпців, типів даних та зв'язків між ними.
Фронтенд	Frontend	"Клієнтська" частина додатку, з якою безпосередньо взаємодіє користувач у своєму веб-браузері.
JSON (Нотація об'єктів	JSON (JavaScript Object	Легкий текстовий формат

JavaScript)	Notation)	для обміну даними, який легко читається як людьми, так і машинами.

Додаток В: Повні лістинги вихідного коду

requirements.txt

```
Flask
psycopg2-binary
python-dotenv
Flask-Cors
```

.env.example (скопіюйте в .env і замініть пароль)

```
DATABASE_URL=postgresql://postgres:your_strong_password@localhost:5432/military_docs
```

run.py

```Python
from app import create_app

app = create_app()

if __name__ == '__main__':
    app.run(debug=True)
```

app/__init__.py

Python

```python
from flask import Flask
from flask_cors import CORS

def create_app():
    app = Flask(__name__)
    CORS(app)

    from. import routes
    app.register_blueprint(routes.api_bp)

    return app
```

app/db.py

Python

```python
import os
import psycopg2
from dotenv import load_dotenv

load_dotenv()

def get_db_connection():
    try:
        conn = psycopg2.connect(os.getenv('DATABASE_URL'))
        return conn
    except psycopg2.OperationalError as e:
        print(f"Помилка підключення до бази даних: {e}")
        return None
```

app/routes.py

Python

```
from flask import Blueprint, jsonify, request
from.db import get_db_connection

api_bp = Blueprint('api', __name__, url_prefix='/api')

@api_bp.route('/docs', methods=)
def get_docs():
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Не вдалося підключитися до бази даних"}), 500

    try:
        with conn.cursor() as cur:
            cur.execute('SELECT id, doc_reference, received_at, source_type, classification, summary,
priority FROM inbound_docs ORDER BY received_at DESC;')
            # Отримуємо назви колонок
            colnames = [desc for desc in cur.description]
            docs = cur.fetchall()

            docs_list =
            for doc in docs:
                doc_dict = dict(zip(colnames, doc))
                # Форматуємо дату в стандартний ISO 8601
                if 'received_at' in doc_dict and doc_dict['received_at']:
                    doc_dict['received_at'] = doc_dict['received_at'].isoformat()
                docs_list.append(doc_dict)
        return jsonify(docs_list)
    except Exception as e:
        return jsonify({"error": str(e)}), 500
    finally:
        if conn:
            conn.close()

@api_bp.route('/docs', methods=)
def add_doc():
    data = request.get_json()

    if not data or not 'doc_reference' in data or not 'summary' in data or not 'classification' in data:
```

```
        return jsonify({"error": "Відсутні обов'язкові поля"}), 400

    doc_ref = data['doc_reference']
    summary = data['summary']
    classification = data['classification']
    source_type = data.get('source_type')
    priority = data.get('priority', 3)

    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Не вдалося підключитися до бази даних"}), 500

    try:
        with conn.cursor() as cur:
            sql = """
              INSERT INTO inbound_docs (doc_reference, summary, classification, source_type, priority)
              VALUES (%s, %s, %s, %s, %s) RETURNING id;
            """
            cur.execute(sql, (doc_ref, summary, classification, source_type, priority))
            new_id = cur.fetchone()
            conn.commit()
        return jsonify({"message": "Документ успішно додано", "id": new_id}), 201
    except Exception as e:
        if conn:
            conn.rollback()
        return jsonify({"error": str(e)}), 500
    finally:
        if conn:
            conn.close()

@api_bp.route('/docs/<int:doc_id>', methods=)
def delete_doc(doc_id):
    conn = get_db_connection()
    if conn is None:
        return jsonify({"error": "Не вдалося підключитися до бази даних"}), 500

    try:
        with conn.cursor() as cur:
            cur.execute("DELETE FROM inbound_docs WHERE id = %s;", (doc_id,))
            if cur.rowcount == 0:
                return jsonify({"error": "Документ з таким ID не знайдено"}), 404
            conn.commit()
        return jsonify({"message": "Документ успішно видалено"}), 200
```

```
    except Exception as e:
        if conn:
            conn.rollback()
        return jsonify({"error": str(e)}), 500
    finally:
        if conn:
            conn.close()
```

frontend/index.html

HTML

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>ІАС Обліку Документів</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Мікро Інформаційно-Аналітична Система</h1>

        <div class="controls">
            <button id="add-doc-btn">Додати документ (тест)</button>
            <input type="text" id="filter-input" placeholder="Фільтрувати за будь-яким полем...">
        </div>

        <table id="docs-table">
            <thead>
                <tr>
                    <th data-column="doc_reference">Референс</th>
                    <th data-column="received_at">Дата отримання</th>
                    <th data-column="source_type">Джерело</th>
                    <th data-column="classification">Гриф</th>
                    <th data-column="summary">Зміст</th>
                    <th data-column="priority">Пріоритет</th>
                    <th>Дії</th>
                </tr>
```

```
            </thead>
            <tbody id="table-body"></tbody>
        </table>
    </div>
    <script src="app.js" defer></script>
</body>
</html>
```

frontend/style.css

CSS

```
body {
    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Helvetica, Arial,
sans-serif;
    background-color: #f4f7f6;
    color: #333;
    margin: 0;
    padding: 20px;
}
.container {
    max-width: 1200px;
    margin: 0 auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
h1 { color: #2c3e50; text-align: center; }
.controls {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
}
#add-doc-btn {
    padding: 10px 15px;
    background-color: #3498db;
    color: white;
    border: none;
```

```
    border-radius: 4px;
    cursor: pointer;
    font-size: 14px;
}
#add-doc-btn:hover { background-color: #2980b9; }
#filter-input {
    padding: 10px;
    width: 300px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 14px;
}
#docs-table { width: 100%; border-collapse: collapse; }
#docs-table th, #docs-table td {
    border: 1px solid #ddd;
    padding: 12px;
    text-align: left;
    vertical-align: top;
}
#docs-table th {
    background-color: #ecf0f1;
    cursor: pointer;
    user-select: none;
}
#docs-table th:hover { background-color: #dbe4e8; }
#docs-table tbody tr:nth-child(even) { background-color: #f9f9f9; }
.delete-btn {
    padding: 5px 10px;
    background-color: #e74c3c;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
.delete-btn:hover { background-color: #c0392b; }
```

frontend/app.js

```
    JavaScript
```

```
const API_URL = 'http://127.0.0.1:5000/api';

const tableBody = document.getElementById('table-body');
const addDocBtn = document.getElementById('add-doc-btn');
const filterInput = document.getElementById('filter-input');
const tableHeaders = document.querySelectorAll('#docs-table th');

let allDocs =;
let sortState = { column: 'received_at', order: 'desc' };

function renderTable(docs) {
    tableBody.innerHTML = '';
    if (docs.length === 0) {
        tableBody.innerHTML = '<tr><td colspan="7" style="text-align: center;">Документи не знайдено.</td></tr>';
        return;
    }
    docs.forEach(doc => {
        const row = document.createElement('tr');
        row.innerHTML = `
            <td>${doc.doc_reference}</td>
            <td>${new Date(doc.received_at).toLocaleString('uk-UA')}</td>
            <td>${doc.source_type |

| ''}</td>
            <td>${doc.classification}</td>
            <td>${doc.summary}</td>
            <td>${doc.priority}</td>
            <td><button class="delete-btn" data-id="${doc.id}">Видалити</button></td>
        `;
        tableBody.appendChild(row);
    });
}

async function fetchDocs() {
    try {
        const response = await fetch(`${API_URL}/docs`);
        if (!response.ok) throw new Error(`HTTP помилка! Статус: ${response.status}`);
        allDocs = await response.json();
        sortAndRender();
    } catch (error) {
        console.error('Не вдалося завантажити документи:', error);
        tableBody.innerHTML = `<tr><td colspan="7" style="text-align: center;">Помилка завантаження даних.</td></tr>`;
```

```
    }
}

function sortAndRender() {
    const { column, order } = sortState;
    const sortedDocs =.sort((a, b) => {
        let valA = a[column], valB = b[column];
        if (column === 'received_at') {
            valA = new Date(valA);
            valB = new Date(valB);
        }
        if (valA < valB) return order === 'asc'? -1 : 1;
        if (valA > valB) return order === 'asc'? 1 : -1;
        return 0;
    });
    updateHeaderArrows();
    filterAndRender(sortedDocs);
}

function filterAndRender(docsToFilter) {
    const filterText = filterInput.value.toLowerCase();
    if (!filterText) {
        renderTable(docsToFilter);
        return;
    }
    const filteredDocs = docsToFilter.filter(doc =>
        Object.values(doc).some(value =>
            String(value).toLowerCase().includes(filterText)
        )
    );
    renderTable(filteredDocs);
}

function updateHeaderArrows() {
    tableHeaders.forEach(th => {
        if (th.dataset.column) {
            const baseText = th.dataset.baseText |

| th.textContent.replace(/ [▾▴]/, '');
            if (!th.dataset.baseText) th.dataset.baseText = baseText;

            if (th.dataset.column === sortState.column) {
                th.textContent = `${baseText} ${sortState.order === 'asc'? '▴' : '▾'}`;
```

```
        } else {
            th.textContent = baseText;
        }
    }
  });
}

addDocBtn.addEventListener('click', async () => {
  const randomId = Math.floor(Math.random() * 1000);
  const newDoc = {
    doc_reference: `TEST-${randomId}`,
    summary: `Це тестовий документ №${randomId}.`,
    classification: 'UNCLASSIFIED',
    source_type: 'OSINT',
    priority: 3
  };
  try {
    const response = await fetch(`${API_URL}/docs`, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(newDoc),
    });
    if (!response.ok) throw new Error(`HTTP помилка! Статус: ${response.status}`);
    alert('Документ успішно додано!');
    fetchDocs();
  } catch (error) {
    console.error('Не вдалося додати документ:', error);
    alert('Помилка при додаванні документа.');
  }
});

tableBody.addEventListener('click', async (event) => {
  if (event.target.classList.contains('delete-btn')) {
    const docId = event.target.dataset.id;
    if (confirm(`Видалити документ з ID ${docId}?`)) {
      try {
        const response = await fetch(`${API_URL}/docs/${docId}`, { method: 'DELETE' });
        if (!response.ok) throw new Error(`HTTP помилка! Статус: ${response.status}`);
        alert('Документ успішно видалено!');
        fetchDocs();
      } catch (error) {
        console.error('Не вдалося видалити документ:', error);
        alert('Помилка при видаленні документа.');
```

```
        }
    }
  }
});

tableHeaders.forEach(header => {
  if (header.dataset.column) {
    header.addEventListener('click', () => {
      const column = header.dataset.column;
      if (sortState.column === column) {
        sortState.order = sortState.order === 'asc'? 'desc' : 'asc';
      } else {
        sortState.column = column;
        sortState.order = 'desc';
      }
      sortAndRender();
    });
  }
});

filterInput.addEventListener('input', () => sortAndRender());

document.addEventListener('DOMContentLoaded', fetchDocs);
```

Додаток С: Зразковий набір даних (sample_data.sql)

SQL

```
INSERT INTO inbound_docs (doc_reference, source_type, classification, summary, analyst_notes, priority) VALUES
('OSINT-24-001', 'OSINT', 'UNCLASSIFIED', 'Аналіз активності в соціальних мережах у регіоні N. Виявлено зростання напруженості.', 'Потребує перехресної перевірки з іншими джерелами.', 2),
('HUMINT-24-001', 'HUMINT', 'SECRET', 'Доповідь про переміщення техніки противника поблизу кордону.', 'Джерело надійне. Дані критичні.', 1),
('SIGINT-24-001', 'SIGINT', 'SECRET', 'Перехоплення радіопереговорів, що вказують на підготовку до провокації.', NULL, 1),
```

('GEOINT-24-001', 'GEOINT', 'CONFIDENTIAL', 'Супутникові знімки показують нові фортифікаційні споруди.', 'Порівняти зі знімками за попередній тиждень.', 2),
('OSINT-24-002', 'OSINT', 'UNCLASSIFIED', 'Стаття в іноземному виданні про економічну ситуацію в країні-агресорі.', 'Перекласти ключові тези.', 3),
('TECH-24-001', 'TECHINT', 'SECRET', 'Аналіз уламків ворожого БПЛА. Виявлено компоненти іноземного виробництва.', 'Передати звіт до технічного відділу.', 1),
('HUMINT-24-002', 'HUMINT', 'CONFIDENTIAL', 'Інформація про настрої серед місцевого населення на окупованих територіях.', 'Джерело потребує додаткової верифікації.', 3),
('OSINT-24-003', 'OSINT', 'UNCLASSIFIED', 'Моніторинг державних закупівель противника. Збільшення замовлень на пально-мастильні матеріали.', NULL, 2),
('SIGINT-24-002', 'SIGINT', 'CONFIDENTIAL', 'Аналіз трафіку стільникового зв''язку в прикордонній зоні.', 'Виявлено аномальну активність.', 2),
('OSINT-24-004', 'OSINT', 'UNCLASSIFIED', 'Публічний звіт міжнародної організації про гуманітарну ситуацію.', 'Використати для підготовки аналітичної довідки.', 3),
('HUMINT-24-003', 'HUMINT', 'SECRET', 'Дані про плани противника щодо проведення інформаційно-психологічної операції.', 'Терміново доповісти керівництву.', 1),
('GEOINT-24-002', 'GEOINT', 'CONFIDENTIAL', 'Знімки з БПЛА, що фіксують замасковані позиції артилерії.', NULL, 1),
('OSINT-24-005', 'OSINT', 'UNCLASSIFIED', 'Аналіз пропагандистських наративів на телебаченні противника за останній місяць.', 'Підготувати звіт про ключові зміни.', 3),
('TECH-24-002', 'TECHINT', 'CONFIDENTIAL', 'Звіт про кібератаку на державні ресурси. Визначено вектор атаки.', 'Потрібен детальний технічний аналіз.', 2),
('HUMINT-24-004', 'HUMINT', 'CONFIDENTIAL', 'Інформація про корупційні схеми у військовому керівництві противника.', 'Дані потребують ретельної перевірки.', 2);

Джерела

1. Python Release Python 3.11.0, доступ отримано серпня 28, 2025, https://www.python.org/downloads/release/python-3110/
2. Installing Python 3.11 on Mac or Windows - PythonTest, доступ отримано серпня 28, 2025, https://pythontest.com/python/installing-python-3-11/
3. Flask Tutorial in Visual Studio Code, доступ отримано серпня 28, 2025, https://code.visualstudio.com/docs/python/tutorial-flask
4. Windows installers - PostgreSQL, доступ отримано серпня 28, 2025, https://www.postgresql.org/download/windows/
5. Downloads - PostgreSQL, доступ отримано серпня 28, 2025, https://www.postgresql.org/download/
6. Installing PostgreSQL on Windows: A Step-by-Step Guide - theDev.uk, доступ отримано серпня 28, 2025, https://thedev.uk/installing-postgresql-on-windows-a-step-by-step-guide/
7. EDB Docs - Installing PostgreSQL on Windows - EnterpriseDB, доступ отримано серпня 28, 2025, https://www.enterprisedb.com/docs/supported-open-source/postgresql/installing/windows/

8. Install PostgreSQL on Windows - GeeksforGeeks, доступ отримано серпня 28, 2025, https://www.geeksforgeeks.org/postgresql/install-postgresql-on-windows/
9. Setup Flask on Windows System using VS Code - DEV Community, доступ отримано серпня 28, 2025, https://dev.to/mursalfk/setup-flask-on-windows-system-using-vs-code-4p9j
10. How To Set Up a Python Virtual Environment on Windows 10 - Liquid Web, доступ отримано серпня 28, 2025, https://www.liquidweb.com/blog/how-to-setup-a-python-virtual-environment-on-windows-10/
11. venv — Creation of virtual environments — Python 3.13.7 documentation, доступ отримано серпня 28, 2025, https://docs.python.org/3/library/venv.html
12. Installation — Flask Documentation (3.1.x), доступ отримано серпня 28, 2025, https://flask.palletsprojects.com/en/stable/installation/
13. virtualenv in PowerShell? - python - Stack Overflow, доступ отримано серпня 28, 2025, https://stackoverflow.com/questions/1365081/virtualenv-in-powershell
14. Create a virtual environment (windows) - Autodesk product documentation, доступ отримано серпня 28, 2025, https://help.autodesk.com/view/SGDEV/ENU/?guid=SGD_jb_jira_bridge_installation_guide_05a_prepare_for_installation_virtual_env_windows_html
15. Department of Defense Metadata Guidance - Chief Digital and Artificial Intelligence Office, доступ отримано серпня 28, 2025, https://www.ai.mil/Portals/137/Documents/Resources%20Page/DoD%20Metadata%20Guidance.pdf
16. U.S. Army Corps of Engineers GUIDELINES AND STANDARDS FOR IMPLEMENTATION OF ELECTRONIC DOCUMENT MANAGEMENT SYSTEMS - National Archives, доступ отримано серпня 28, 2025, https://www.archives.gov/files/records-mgmt/toolkit/pdf/ID114.pdf
17. Types of Intelligence Analysis - Augusta University, доступ отримано серпня 28, 2025, https://www.augusta.edu/online/blog/types-of-intelligence-analysis
18. Project Layout — Flask Documentation (3.1.x), доступ отримано серпня 28, 2025, https://flask.palletsprojects.com/en/stable/tutorial/layout/
19. Build a Scalable Flask Web Project From Scratch - Real Python, доступ отримано серпня 28, 2025, https://realpython.com/flask-project/
20. Welcome to Flask — Flask Documentation (3.1.x), доступ отримано серпня 28, 2025, https://flask.palletsprojects.com/
21. Preventing SQL Injection Attacks With Python, доступ отримано серпня 28, 2025, https://realpython.com/prevent-python-sql-injection/
22. Passing parameters to SQL queries - psycopg 3.3.0.dev1 documentation, доступ отримано серпня 28, 2025, https://www.psycopg.org/psycopg3/docs/basic/params.html
23. psycopg2 and SQL injection security - python - Stack Overflow, доступ отримано серпня 28, 2025, https://stackoverflow.com/questions/45128902/psycopg2-and-sql-injection-security
24. Flask-Cors 3.0.10 documentation, доступ отримано серпня 28, 2025,

https://flask-cors.readthedocs.io/en/latest/
25. Tackle Cross-Origin Resource Sharing with the Flask-CORS Extension - Apidog, доступ отримано серпня 28, 2025, https://apidog.com/blog/flask-cors/
26. Quickstart: PostgreSQL Extension for VS Code | Microsoft Learn, доступ отримано серпня 28, 2025, https://learn.microsoft.com/en-us/azure/postgresql/extensions/vs-code-extension/quickstart-connect
27. PostgreSQL - Visual Studio Marketplace, доступ отримано серпня 28, 2025, https://marketplace.visualstudio.com/items?itemName=ms-ossdata.vscode-pgsql
28. Managing Postgres Directly in VS Code - Neon, доступ отримано серпня 28, 2025, https://neon.com/blog/managing-postgres-directly-in-vs-code
29. How To: Install Live Server in VS Code - Code:You | Free Tech Training, доступ отримано серпня 28, 2025, https://code-you.org/students/resources/guides/install-live-server-in-vs-code/
30. Live Server - Visual Studio Marketplace, доступ отримано серпня 28, 2025, https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer
31. How to Enable Live Server on Visual Studio Code - GeeksforGeeks, доступ отримано серпня 28, 2025, https://www.geeksforgeeks.org/installation-guide/how-to-enable-live-server-on-visual-studio-code/
32. Terminal Basics - Visual Studio Code, доступ отримано серпня 28, 2025, https://code.visualstudio.com/docs/terminal/basics
33. Running Python code in Visual Studio Code, доступ отримано серпня 28, 2025, https://code.visualstudio.com/docs/python/run
34. Getting Started with Python in VS Code, доступ отримано серпня 28, 2025, https://code.visualstudio.com/docs/python/python-tutorial
35. Getting Started with the PostgreSQL Extension for VSCode | by Ryan Hutzley | Medium, доступ отримано серпня 28, 2025, https://ryanhutzley.medium.com/getting-started-with-the-postgresql-extension-for-vscode-d666c281ec72
36. How to Connect to PostgreSQL From Visual Studio Code - CommandPrompt Inc., доступ отримано серпня 28, 2025, https://www.commandprompt.com/education/how-to-connect-to-postgresql-from-visual-studio-code/
37. What is the PostgreSQL extension for Visual Studio Code preview?, доступ отримано серпня 28, 2025, https://learn.microsoft.com/en-us/azure/postgresql/extensions/vs-code-extension/overview
38. Python in Visual Studio Code, доступ отримано серпня 28, 2025, https://code.visualstudio.com/docs/languages/python
39. Python environments in VS Code, доступ отримано серпня 28, 2025, https://code.visualstudio.com/docs/python/environments