

Streamlit + SQLite + GPT מיני פרויקט LegalSmart

פרויקט שמדגים איך בונים אפליקציית ניהול נתונים עם ניתוח חכם בעזרת GPT

שלב 1 – התקנת הספריות הדרשיות לפרויקט

משימה 1.1 – התקנת חבילות

כתב את הפקודה הבאה בטרמינל:

```
pip install streamlit pandas openai python-dotenv
```

ודא שאין שגיאות בהתקנה. אם יש – קרא את ההודעה בעין וחפש פתרון (או שאל אותה).

משימה 1.2 – יצירת קובץ Python

צור קובץ חדש בשם:

`main.py`

והוסף אליו את שורות ה- `import` מהתמונה שלך:

```
import streamlit as st
import sqlite3
import pandas as pd
import openai
import os
from dotenv import load_dotenv
```

סיכום החבילות שהתקנו ולמה:

למה צריך אותה?	ספרייה
מסך אינטראקטיבי להצגת הטופס	streamlit
הצגת הטבלה של הלקוחות	pandas
שמירת נתונים מקומית	sqlite3
חיבור ל-GPT	openai
קריאה של מפתח API מקובץ חיצוני	python-dotenv

שלב 2 – יצרת קובץ `env`. לשימרת מפתח API

מטרה: לא לשים מפתחות סודים בקובד עצמו, אלא בקובץ נפרד שנקרא `env`.

משימה 2.1 – קבלת מפתח API

1. היכנס לאתר <https://platform.openai.com/account/api-keys>
2. אם אין לך מפתח – API לחץ על Create new secret key
3. העתק את המפתח – הוא ייראה כמו מחרוזת שמתחליה ב-...-sk
4. שמור אותו זמנית לצד (פתקן/טקסט) – לא תוכל לראותו שוב

משימה 2.2 – יצרת קובץ `env`.

בתיקיית הפרויקט (ליד `main.py`) צור קובץ חדש בשם:

`.env`

זה צריך להיראות כך:



שים לב: שם הקובץ מתחילה בנקודה – זה חשוב `env`. ולא `env.txt`.

משימה 2.3 – כתיבה לתוך הקובץ

בתוך הקובץ `env`. כתוב את השורה הבאה (החלף את הערך במפתח שלך):

`OPENAI_API_KEY=sk-xx`

חשיבות: אין מרווחים, אין גרשים. רק שם המשתנה = ואז הערך שלו.

階段 3 – טיענת מפתח ה API-לקיים והכנה לשימוש עמו GPT 🔑

מטרה: לגרום לכך שהקוד ידע להשתמש ב מפתח שלך כדי לדבר עם ChatGPT
נשתמש בספריות `os.getenv` כדי לקרוא את המפתח מתוך קובץ `env`.

משימה 3.1 – וודא שיבאת את הספריות הדרשיות

בקובץ `uk_main.py` בחלק העליון, אמורות להופיע (בין השאר) השרוטות הבאות:

```
import openai
import os
from dotenv import load_dotenv
```

אםichert מהן חסירה – תוסיף אותה.

משימה 3.2 – טען את קובץ `env`.

מיד אחרי שורות ה-`import` כתוב את השרורה הזו:

```
load_dotenv()
```

מה זה עושה? 🔎

זו פקודה שאומرت לפיתון: “קרא את הקובץ `env`. ושים את המשתנים שבו בתוך – `os.environ`. כלומר, תוכל לגשת אליהם בקוד.

משימה 3.3 – שמור את המפתח בתוך `key_api`

הווסף את השרורה:

```
openai.api_key = os.getenv("OPENAI_API_KEY")
```

מה זה עושה? 🔎

- `os.getenv("OPENAI_API_KEY")` מוחפש משתנה בשם זהה מתוך הקובץ `env`.
- אם נמצא – הוא שולח אותו ל- OpenAI כדי שתוכל להשתמש בו לבקשת ל-GPT

שלב 4 – ייצרת בסיס נתונים (clients.db) עם טבלה ללקוחות

- ☞ מטרה: ליצור קובץ נתונים שייחסן את רשותה של משרד עורכי הדין
☞ הפעלה תיקרא clients ובה נומרו: מזהה, שם, גיל, וסוג הפניה המשפטית
-

משימה 4.1 – הגדרת פונקציה בשם (init_db) (אפשר להשתמש בתרגיל בית הקודם)

1. כתוב פונקציה בשם init_db

2. הפונקציה צריכה:

- להתחבר לקובץ בשם clients.db אם לא קיים – יוצר אוטומטית
- ליצור טבלה בשם clients עם 4 שדות:

```
id INTEGER PRIMARY KEY AUTOINCREMENT,  
name TEXT,  
age INTEGER,  
legal_issue TEXT
```

טייפ: תשתמש ב- SQL בסיסי עם הפקודה CREATE TABLE IF NOT EXISTS

משימה 4.2 – הוספה נתוני דוגמה (רק אם הפעלה ריקה) (אפשר להשתמש בתרגיל בית הקודם)

1. הוסף רשימת לקוחות לדוגמה

2. כל לקוח צריך לכלול:

- שם
- גיל
- תחום משפטי (אחד מ: משפחה, נדל"ן, חוות, פלילי, צוואות וירושות)

הכנס לפחות 10 לקוחות לדוגמה (אפשר בעברית, כולל גיל ותחום). אפשר להשתמש בזיהויים:

```
] = sample_data  
,("David Levi", 42, "Real Estate")  
,("Noa Cohen", 35, "Family")  
,("Itamar Ben-Ari", 60, "Wills and Inheritance")  
,("Yael Mizrahi", 29, "Contracts")  
,("Avi Dahan", 45, "Criminal")  
,("Rina Azulay", 38, "Family")  
,("Daniel Kadosh", 50, "Wills and Inheritance")  
,("Lior Avrahami", 33, "Real Estate")  
,("Maya Segal", 41, "Family")
```

("Eliad Shlomo", 28, "Contracts")

[

.connection.commit() לא לשכוח לעשות commit ולסגור את ה-connection.

■ **שלב 5 – יצירת פונקציה להוספה לקוח חדש לatable**

מטרה: לאפשר לתוכנית להוסיף לקוח חדש (שם, גיל, תחום משפטי) אל בסיס הנתונים

• **משימה 5.1 – כתיבת פונקציה בשם add_client**

- הפונקציה מקבלת 3 פרמטרים:
 1. name שם הלקוח
 2. age גיל הלקוח
 3. issue תחום המשפט של הפניה

• **משימה 5.2 – התחברות לבסיס הנתונים**

- בטור הפונקציה, התחבר לקובץ dbclients.db
- השתמש בפונקציה sqlite3.connect(...) כדי ליצור את החיבור
- צור גם – ((Cursor (conn.cursor() מרים SQL

• **משימה 5.3 – הרצת פקודת SQL להוספה**

- כתוב פקודה ? ? ? VALUES (...) (, , ,
INSERT INTO clients (name, age, legal_issue) VALUES (?, ?, ?,
, , ,
(name, age, issue))
- השתמש ב- ? כסימנים למילוי פרמטרים, אז תן את הערכים המתאימים ב-tuple. זה צריך להיראות כך:

```
INSERT INTO clients (name, age, legal_issue) VALUES (?, ?, ?,  
(name, age, issue)
```

הסיבה שימושים ב- ? ולא במחרוזת טקסט רגילה היא כדי להגן מפני טעויות ופרצונות אבטחה. (SQL Injection)

• **משימה 5.4 – שימירת השינויים וסגירת החיבור**

- אחרי הפקודה, כתוב conn.commit() כדי לשמור את ההוספה
- ואז conn.close() כדי לסגור את החיבור

שלב 6 – שליפת כל הנתונים מהטבלה (load_data)

 **מטרה:** לכתוב פונקציה שמחזירה את כל הנתונים מבוסיס הנתונים clients.db dataframe pandas כລונר

משימה 6.1 – כתיבת שם הפונקציה

- צור פונקציה בשם load_data
- הפונקציה לא מקבלת שום פרמטר
- בסוף הפונקציה היא תחזיר את הטבלה כ- DataFrame של pandas

משימה 6.2 – התחברות לבסיס הנתונים

- התחבר לקובץ db.clients בדיק כמו בשלבים הקודמים
- צור Cursor אם אתה רוצה

משימה 6.3 – הרצת שאלחת SQL עם pandas

- השתמש בפונקציה של pandas שנקראת:

`pd.read_sql_query(...)`

- קרא לה עם השאילתה:

`SELECT * FROM clients`

- עם החיבור שיצרת (conn)

 **התוצאה שתחזיר היא טבלה מסווג – pandas DataFrame** מבנה מאוד נוח להציג וניתוח של נתונים.

משימה 6.4 – סגירת החיבור והחזרת התוצאה

- כתוב `()close.conn` כדי לסגור את הקשר לבסיס הנתונים
- החזיר את הטבלה שיצרת מהפונקציה עם return

שלב 7 – שימוש ב-GPT-לניתוח טבלת נתונים

 **מטרה:** להשתמש ב-GPT באמצעות API של OpenAI (OpenAI) כדי לחת ניתוח אוטומטי של הנתונים
שייש לנו בטבלה

```
def analyze_data(df):
    from openai import OpenAI
    client = OpenAI()

    prompt = f"""
        הנה טבלת נתונים של לקוחות משרד עורכי דין
        {df.to_markdown()}

        תן לי ניתוח של
        מהו הגיל הממוצע
        1. אילותחוםים חזזריים יותר?
        2. האם קיימת נטייה לדפוס גיל לפני תחום משפטי
        """

    response = client.responses.create(
        model="gpt-4.1",
        input=prompt
    )
    return response.output_text
```

(אבקש מכם לקרוא מה הקוד עשו, אתם עוד תרchieבו על זה במודול של LLM)

שלב 8 – בוט חכם ששולש שאלות על הנתונים

 **מטרה:** לאפשר למשתמש להקליד שאלה חופשית (למשל: מה התחום הכי נפוץ?)
GPT יקבל את השאלה ו咎 את הנתונים וינסה לענות עליה בצורה מדוקقة.

```

# --- Chatbot on data ---
def chatbot_response(user_input, df):
    prompt = f"""
{ענה על השאלה הבאה בהתאם על הנתונים של משפט עורך הדיון:
{df.to_markdown()}

{user_input}
"""

from openai import OpenAI
client = OpenAI()
response = client.responses.create(
    model="gpt-4.1",
    input=prompt
)
return response.output_text

```

 **שלב 9 – בניית טופס להזנת ל��וחות עם Streamlit (משלב זה הניסו להשלים את הקוד בהתאם לשלבים המתוארים)**

 **מטרה:** לאפשר למשתמש להזין לקוח חדש דרך טופס אינטרנט ולשמור אותו בדatabe יי

שלב 9.1 – קריאה לפונקציית ייצרת הדatabe יי

init_db()

 **מה זה עושים?**

- מבטיח שלפני שהממשק מוצג, יוצר בסיס הנתונים (אם הוא עוד לא קיים).
- אם זו הפעם הראשונה שהמשתמש פותח את המערכת – זה יודא שיש טבלה מוכנה עם נתונים.

 **תמיד נקרא בשורה הראשונה של האפליקציה.**

שלב 9.2 – ייצרת כותרת ראשית לאפליקציה

```
st.title(" LegalSmart - ניהול ל��וחות משפטיים")
```

 **מה זה עושים?**

- מציג כותרת צבעונית בראש הדף – כמו שם מערכת.
- הסמל  מוסיף מראה מקצועית.

שלב 9.3 – כוורתה משנה לטופס ההוספה

```
st.header("+" ("הוסף ללקוח חדש"))
```

מה זה עושה? 

- מוסיף כוורתה שנייה (קטנה יותר) מעל הטופס
- מצין שזה החלק שבו מוסיפים ללקוח חדש

שלב 9.4 – התחלת טופס חדש (עם מזהה)

```
with st.form("new_client_form"):
```

מה זה עושה? 

- يוצר טופס שאפשר למלא אותו ואז להחזיר "שמור"
- הוא מזהה פנימי לטופס (אפשר לקרוא לו איר שרטוטים)

שלב 9.5 – יצירת שדות קלט בטופס

שם הלקוח: 

```
name = st.text_input("שם")
```

- שדה קלט טקסטואלי – מקבלים שם בתור מחזרות
- נשמר במשתנה name

גיל הלקוח: 

```
age = st.number_input("גיל", min_value=18, max_value=120)
```

- שדה קלט מספרי עם מגבלת גיל
- מנע הכנסת גילאים לא סבירים
- נשמר במשתנה age

תחום משפטי: 

```
issue = st.selectbox("סוג פניה", ["פלילי", "חוויים", "נדיל", "צוואות וירושות", "משפחה"])
```

- תיבת בחירה עם ערכים קבועים (dropdown)
- המשתמש בוחר רק מתוך האפשרויות הקיימות
- נשמר במשתנה issue

שימוש ב-selectbox מונע טעויות כתיב מצד המשתמש. 

שלב 9.6 – כפתור "שמור" וטיפול בלחיצה

```
submitted = st.form_submit_button("שמור")
```

- יוצר כפתור בסוף הטופס
- הtoutזאה שלו נשמרת במשתנה submitted
- אם המשתמש לחץ עליו – המשתנה יהיה True

```
if submitted:
```

```
    add_client(name, age, issue)  
    st.success("ل寇ח נוסף בהצלחה!")
```

אם המשתמש לחץ:

- הפקנץיה (…)(...)add תוסיף את הל寇ח לדאטביבו
- תוצג הודעה הצלחה

מואוד חשוב: כל קלטים + כפתור חייכים להיות בתוך הבלוק של `(...)`with `st.form`

שלב 10 – הצגת טבלת כל הפניות באתר (View Table)

מטרה: להציג את כל הנתונים שנשמרו בסיס הנתונים, בטבלה ברורה ונוחה לצפייה בתוך הדף של האפליקציה

שלב 10.1 – הצגת כותרת חלק

```
st.header("כל הפניות")
```

מה זה עושה?

- מוסיף כותרת ברורה לדף שמצוינת שהטבלה שתוצג תוצג היא של כל הפניות הקיימות

שלב 10.2 – שיליפת הנתונים

```
df = load_data()
```

מה זה עושה?

- קורא לפונקציה `load_data()` ששולפת את כל הנתונים מהטבלה clients
- התוצאה היא טבלה מסווג `DataFrame` מבנה שמכיל שורות ועמודות כמו Excel
- נשמר לתוכה משתנה בשם `df` קיזור של `DataFrame`

הערה: הקוד הזה לא טוען את הטבלה מחדש בזמן אמת – הוא טוען רק כשמריצים את האפליקציה או כשמרעננים את הדף.

שלב 10.3 – הציגת הטבלה במסך

```
st.dataframe(df)
```

מה זה עושה? 

- מציג את הנתונים של df בטבלה שנייתן לגלול, למין ולסדר לפי עמודות
 - התצוגה מאוד נוחה למשתמש
- יתרונ: ברגע שהוזן לקוח חדש בטופס – ברגען של הדף, הוא מיד יופיע כאן.

שלב 11 – ניתוח הנתונים בעזרת GPT בלחיצת כפתור

מטרה: לאפשר למשתמש ללחוץ על כפתור, לשЛОח את כל הטבלה ל-GPT ולקבל ניתוח חכם אוטומטי של הנתונים

שלב 11.1 – כוורת לחلك של ניתוח

```
st.header("📝 ניתוח בעזרת GPT")
```

מה זה עושה? 

- מציג כוורת שמודיעה למשתמש: כאן תבוצע פעולה חכמה
- הסמל  מדגיש שהזה קשור לניתוח נתונים

שלב 11.2 – יצירת כפתור “בצע ניתוח”

```
if st.button("בצע ניתוח"):
```

מה זה עושה? 

- يוצר כפתור שعلוי כתוב “בצע ניתוח”
- כשמשתמש לוחץ עליו, הקוד שמתוחת לשורה זו יירץ
- כל מה שנמצא בתוך ה- if קורה רק אם הכפתור נלחץ

שלב 11.3 – קריאה לפונקציית ניתוח הנתונים

```
analysis = analyze_data(df)
```

מה זה עושה? 

- שולח את כל הנתונים (df) לפונקציה שכתבתם קודם analyze_data(df) קודם
- בפניהם היא ייצור prompt ששולח ל-GPT את הטבלה, מקבלת תובנות, ומחייב טקסט התוצאה נשמרת במשתנה analysis ככלומר: התשובה של GPT

זו השורה שמבצעת את “הקסם” מאחורי הקלעים. 

● שלב 11.4 – הציגת הניתוח על המסר

```
st.markdown(analysis)
```

.ma זה עושה?

- מציג את טקסט התוצאה מה ש-GPT החזיר
- כיוון שהוא טקסט בפורמט Markdown אפשר להשתמש בו עם נתני כתורות, רשימות, טבלאות ועוד

● שלב 12 – צ'אט-בוט שמאפשר לשאול שאלות על הנתונים

🎯 מטרה: המשתמש מקליד שאלה פתוחה על הנתונים (למשל: מה התחום הכי נפוץ?) GPT מקבל את השאלה והנתונים – ועונה תשובה מותאמת

● שלב 12.1 – כוורת לחילוק הצ'אט-בוט

```
st.header("🤖 שאל את LegalSmart")
```

ma זה עושה?

- מציג כוורת של החלק שבו המשתמש ישאל את הבוט
- הסמל 🤖 מדגיש שגם אינטראקציה עם בינה מלאכותית

● שלב 12.2 – שדה טקסט להזנת שאלה

```
user_q = st.text_input("שאלה על הנתונים")
```

ma זה עושה?

- מציג שדה שבו המשתמש יכול להקליד שאלה בעברית (או באנגלית)
- התוצאה נשמרת במשתנה user_q

חשוב: אין בדיקה על תקינות השאלה – אפשר לשאול כל דבר, כל עוד יש קשר לננתונים

שלב 12.3 – כפתור לשילוח השאלה

```
if st.button("שלח"):
```

מה זה עושה? 

- יוצר כפתור בשם "שלח"
- אם לוחצים עליו – התנאי יתבצע

שלב 12.4 – בדיקה שהשדה לא ריק

```
if user_q.strip():
```

מה זה עושה? 

- מודד אם השדה שהזונה לא ריק
- strip מסיר רווחים מההתחלת והסוף
- אם המשתמש לא כתב כלום – הקוד שמתוחת לא ירוץ

שלב 12.5 – שליחת השאלה ל- GPT ומצגת התשובה

```
response = chatbot_response(user_q, df)  
st.markdown(response)
```

מה זה עושה? 

- קורא לפונקציה chatbot_response(...) שכתבתם קודם
- היא מקבלת גם את השאלה (user_q) וגם את כל הנתונים (df)
- התשובה חוזרת בטקסט ומוצגת באמצעות st.markdown(...)

כל שנותר זה להפעיל את האפליקציה:

streamlit run main.py