für diese route: @app.route("/create_dog", methods=["GET", "POST"]) def create_dog(): if request.method == "POST": if "username" not in session: return redirect(url_for("profile")) username = session["username"] dog_name = request.form.get("name") breed = request.form.get("breed") age_years = request.form.get("age_years") weight_kg = request.form.get("weight_kg") sensitivities = request.form.get("sensitivities") conn = get_db_connection() cur = conn.cursor() füge cur.execute hinzu

So könntest du **cur.execute** direkt ergänzen (einfach und ohne Extras):

```
cur.execute(
    "INSERT INTO dogs (username, name, breed, age_years, weight_kg, sensitivities) VALUES (%s, %s, %s, %s, %s, %s)",
    (username, dog_name, breed, age_years, weight_kg, sensitivities)
)
conn.commit()
```

Wenn deine Tabelle oder Spalten anders heißen, sag kurz Bescheid, dann passe ich es exakt an.

in plans_all stellt man sich sein abo-modell zusammen. bitte ein ähnliches route.

Alles klar 🙂
Hier ein **ähnliches Flask-Route-Beispiel** zu create_dog, nur eben für **plans_all**, wo sich ein User sein Abo-Modell zusammenstellt.

Ich halte es bewusst **einfach und im gleichen Stil** wie deine bestehende Route:

```
@app.route("/plans_all", methods=["GET", "POST"])
def plans_all():
    if request.method == "POST":
        if "username" not in session:
            return redirect(url_for("profile"))

        username = session["username"]

        plan_name = request.form.get("plan_name")
        duration_months = request.form.get("duration_months")
        price = request.form.get("price")
```

```
    features = request.form.get("features")


    conn = get_db_connection()
    cur = conn.cursor()


    cur.execute(
      "INSERT INTO plans (username, plan_name, duration_months, price, features)
VALUES (%s, %s, %s, %s, %s)",
      (username, plan_name, duration_months, price, features)
    )


    conn.commit()
    cur.close()
    conn.close()


    return redirect(url_for("profile"))


  return render_template("plans_all.html")
```

Annahmen

- Tabelle heißt plans
- Abo ist **usergebunden** (username)
- Formularfelder:
    - plan_name
    - duration_months
    - price
    - features

Wenn:

- mehrere Optionen auswählbar sind (Checkboxen)
- oder ein bestehendes Abo **überschrieben** werden soll

- oder es eine **Plan-ID** statt username gibt

→ sag kurz Bescheid, dann passe ich es exakt an deine Logik an.

wie heißt diese praktik in python: dog_name = request.form.get("name") breed = request.form.get("breed") age_years = request.form.get("age_years") weight_kg = request.form.get("weight_kg") sensitivities = request.form.get("sensitivities")

Das nennt man **Formulardaten auslesen** – genauer gesagt:

✅ **Auslesen von POST-Formulardaten (Form Data Access)**

In Python mit Flask spricht man meist von:

- **Zugriff auf request.form**
- oder **Form Data Retrieval**

Technisch passiert hier Folgendes:

- request.form ist ein **dict-ähnliches Objekt**
- .get("feldname") ist **Dictionary-Zugriff mit Fallback (None)**

Etwas formeller ausgedrückt:

Zuweisung von Formularfeldern zu Variablen mittels Dictionary-Lookup

Kurz & praxisnah:

- ❌ kein spezielles Design Pattern
- ✅ einfaches **Form Handling in Flask**
- ✅ Nutzung von **.get() zur sicheren Feldabfrage**

Alternative Bezeichnungen, die du hören könntest:

- *Form Parsing*
- *Request Data Extraction*
- *Mapping von Formularfeldern auf Variablen*

Wenn du willst, kann ich dir auch zeigen:

- wie man das **kürzer** schreibt
- oder wie man es **validiert** (leer / Typen)
- oder wie man es **dynamisch** löst ohne jede Zeile einzeln zu schreiben