Subscription cancellation feature implementation

Dies ist eine Kopie eines Chats zwischen Claude und Tim. Der Inhalt kann nicht verifizierte oder unsichere Inhalte enthalten, die nicht die Ansichten von Anthropic widerspiegeln. Der geteilte Snapshot kann Anhänge und Daten enthalten, die hier nicht angezeigt werden.

def estimate_grams_per_day(weight_kg): try: w = float(weight_kg) if weight_kg is not None and str(weight_kg).strip() != "" else None w = float(weight_kg) except: w = None return max(50, int(w * 20)) if w else 200 def get_product_by_category(cur, category_name):

pasted

app.route("/subscriptions/<int:sub_id>/cancel", methods=["POST"])def cancel_subscription(sub_id): if "username" not in session: return redirect(url_for("profile")) username = session["username"] conn = get_db_connection() cur = conn.cursor() cur.execute(""" DELETE FROM subscriptions WHERE id = ? AND username = ? """, (sub_id, username)) conn.commit() conn.close() return redirect(url_for("manage_subscriptions"))@app.route("/cart/add", methods=["POST"])def cart_add(): Collapse filepython-sql-app/dogtreats.dbCopy file name to clipboard0 BytesBinary file not shown. Collapse filepython-sql-app/templates/manage_subscriptions.htmlCopy file name to clipboardExpand all lines: python-sql-app/templates/manage_subscriptions.html+7Lines changed: 7 additions & 0 deletionsOriginal file line numberDiff line numberDiff line change@@ -2,6 +2,7 @@{% block content %}<h1>Abos verwalten</h1>{% for sub in subscriptions %} <h3>{{ sub.plan_title }} – {{ sub.dog_name }}</h3>@@ -39,6 +40,12 @@ <h3>{{ sub.plan_title }} – {{ sub.dog_name }}</h3> {% endif %}</form><form method="post" action="{{ url_for('cancel_subscription', sub_id=sub.id) }}" onsubmit="return confirm('Abo wirklich kündigen?');"> <button type="submit" class="btn-danger">Abo kündigen</button> </form> {% endfor %}

Weniger anzeigen

Looking at this code, I can see several issues that need fixing:

Problems Found

1. **Duplicate function definitions**: build_plan() and recommend_plans() are both defined twice with different signatures
2. **Inconsistent variable naming**: The first build_plan() uses product_row but references product
3. **Missing function implementation**: First build_plan() has an incomplete body

4. **Logic conflicts**: The two versions of recommend_plans() have completely different approaches

Fixed Code

Here's the corrected version:

python

```
def estimate_grams_per_day(weight_kg):
  try:
    w = float(weight_kg) if weight_kg is not None and str(weight_kg).strip() != "" else None
  except:
    w = None
  return max(50, int(w * 20)) if w else 200


def get_product(cur, category_name, exclude_keyword=None):
  if exclude_keyword:
    cur.execute("""
      SELECT p.name, p.description, p.base_amount_g, p.base_price_eur, p.image_url,
          c.name AS category_name
      FROM products p
      JOIN categories c ON p.category_id = c.id
      WHERE c.name = ?
       AND LOWER(p.name) NOT LIKE ?
      ORDER BY p.id ASC
      LIMIT 1
    """, (category_name, f"%{exclude_keyword.lower()}%"))
  else:
    cur.execute("""
      SELECT p.name, p.description, p.base_amount_g, p.base_price_eur, p.image_url,
          c.name AS category_name
      FROM products p
```

```
        JOIN categories c ON p.category_id = c.id
        WHERE c.name = ?
        ORDER BY p.id ASC
        LIMIT 1
    """, (category_name,))
  return cur.fetchone()

def build_plan(plan_id, title, product, note, tags, grams_per_day):
  if not product:
    return None
  return {
    "plan_id": plan_id,
    "title": title,
    "note": note,
    "product_name": product["name"],
    "product_desc": product["description"],
    "category_name": product["category_name"],
    "base_amount_g": int(product["base_amount_g"]),
    "base_price_eur": float(product["base_price_eur"]),
    "image_url": product["image_url"],
    "grams_per_day": grams_per_day,
    "tags": tags
  }

def recommend_plans(cur, dog):
  age = dog["age_years"]
  sens = (dog["sensitivities"] or "").lower()

  is_puppy = age is not None and age <= 1
```

```
is_senior = age is not None and age >= 8

grams_per_day = estimate_grams_per_day(dog["weight_kg"])

catalog = []

catalog.append(build_plan(
    "base",
    "Basis-Abo",
    get_product(cur, "Trockenfutter"),
    f"Solide Grundversorgung für {dog['name']}.",
    ["base"],
    grams_per_day
))

catalog.append(build_plan(
    "grainfree",
    "Getreidefrei-Abo",
    get_product(cur, "Getreidefrei"),
    "Ohne Getreide – gut verträglich für sensible Hunde.",
    ["getreide"],
    grams_per_day
))

catalog.append(build_plan(
    "sensitive",
    "Sensitiv-Magen-Abo",
    get_product(cur, "Sensitiv"),
    "Besonders schonend für empfindlichen Magen.",
```

```
    ["magen", "sensitiv"],
    grams_per_day
))

catalog.append(build_plan(
    "senior",
    "Senior-Abo",
    get_product(cur, "Sensitiv"),
    "Leicht verdaulich und angepasst an ältere Hunde.",
    ["senior"],
    grams_per_day
))

catalog.append(build_plan(
    "junior",
    "Junior-/Welpen-Abo",
    get_product(cur, "Trockenfutter"),
    "Für Wachstum und hohen Energiebedarf.",
    ["junior"],
    grams_per_day
))

catalog.append(build_plan(
    "rindfrei",
    "Rindfrei-Abo",
    get_product(cur, "Trockenfutter", "rind"),
    "Ohne Rind – bei entsprechender Unverträglichkeit.",
    ["rind"],
    grams_per_day
```

```
))

catalog.append(build_plan(
    "huhnfrei",
    "Huhnfrei-Abo",
    get_product(cur, "Trockenfutter", "huhn"),
    "Ohne Huhn – geeignet bei Allergien.",
    ["huhn"],
    grams_per_day
))

catalog.append(build_plan(
    "fischfrei",
    "Fischfrei-Abo",
    get_product(cur, "Trockenfutter", "fisch"),
    "Ohne Fisch – für empfindliche Hunde.",
    ["fisch"],
    grams_per_day
))

catalog.append(build_plan(
    "low_energy",
    "Low-Energy-Abo",
    get_product(cur, "Trockenfutter"),
    "Reduzierte Energiezufuhr für ruhige Hunde.",
    ["low_energy"],
    grams_per_day
))
```

```
catalog.append(build_plan(
    "chew",
    "Kauartikel-Abo",
    get_product(cur, "Kauartikel"),
    "Kauartikel zur Beschäftigung – Ergänzung zum Futter.",
    ["chew"],
    grams_per_day
))

catalog = [c for c in catalog if c]

def score(p):
    s = 0
    if "getreideunverträglichkeit" in sens and "getreide" in p["tags"]:
        s += 50
    if "empfindlicher magen" in sens and "magen" in p["tags"]:
        s += 45
    if "rind-unverträglichkeit" in sens and "rind" in p["tags"]:
        s += 60
    if "huhn-unverträglichkeit" in sens and "huhn" in p["tags"]:
        s += 60
    if "fisch-unverträglichkeit" in sens and "fisch" in p["tags"]:
        s += 60
    if "niedriger energiebedarf" in sens and "low_energy" in p["tags"]:
        s += 40
    if is_senior and "senior" in p["tags"]:
        s += 40
    if is_puppy and "junior" in p["tags"]:
        s += 40
```

```
    if "base" in p["tags"]:
        s += 10
    return s

catalog.sort(key=score, reverse=True)

return catalog[:3]
```

Key Changes

1. **Removed duplicate definitions** - kept the more complete versions
2. **Fixed estimate_grams_per_day()** - removed redundant line
3. **Removed get_product_by_category()** - consolidated into get_product()
4. **Standardized on the better recommend_plans() version** - the one with scoring logic and tag-based matching