perplexity

Why do neither of the flask functions correctly output the data?

```
#Vals recieves: "input1_type","input1" etc etc
    count=1 #Counter for while loop
    while f"input{count}_type" in vals: #Run through all the input types in vals
        type=vals[f"input{count}_type"] #Gets whatever the variable type is
        user_input=vals[f"input{count}"] #Gets whatever the input from the user was
        submit_data[reference.index(type)]=user_input #Finds where the type is inside reference and stores the user input in the same spot in the data submission variable
        count+=1 #iterates the
    # Try 2
    for key in vals.keys():
        try:
            ind = reference.index(key)
            submit_data[ind] = vals.get(key)
        except ValueError:
            continue
    warnings = submit_patient_data(submit_data)
```

Both snippets are indexing `reference` with the wrong thing, so they never put values where you expect.

How the first loop misbehaves

```
count = 1
while f"input{count}_type" in vals:
    type = vals[f"input{count}_type"]
    user_input = vals[f"input{count}"]
    submit_data[reference.index(type)] = user_input
    count += 1
```

- `vals` has keys like `"input1_type"` and `"input1"`, not the type name itself.
- `reference` is a list of type names (for example: `["input1_type", "input2_type", ...]` or maybe `["age", "height", ...]`).
- `submit_data[reference.index(type)]` only works if `vals[f"input{count}_type"]` is an element of `reference`. If `type` is something like `"text"` or `"number"` but `reference` contains `"input1_type"`, `"input2_type"`, `reference.index(type)` will raise `ValueError` or store in the wrong index.[1]

So: the first loop expects `vals[f"input{count}_type"]` to match exactly an element in `reference`, which is probably not true.

How the “Try 2” loop misbehaves

```
for key in vals.keys():
    try:
        ind = reference.index(key)
        submit_data[ind] = vals.get(key)
    except ValueError:
        continue
```

- `key` runs over `"input1_type"`, `"input1"`, `"input2_type"`, `"input2"`, etc.
- `reference.index(key)` will only succeed if the literal string `"input1"` or `"input1_type"` is inside `reference`.[1]
- If `reference` instead stores semantic names or type names (e.g., `"age"`, `"height"`, `"weight"`), the `index` call always fails and you hit `ValueError`, so no entries are ever written into `submit_data`.

In other words, in the first attempt you index `reference` by the *value* from `vals[...]`, and in the second you index `reference` by the *key* (`"input1"`, etc.), but `reference` does not contain either of those.

What to change

Decide what `reference` actually contains and consistently match that:

- If `reference` is a list of the **type values** stored in `vals["inputX_type"]`, then the first loop is structurally right but you must ensure those type values are actually present in `reference`:

  ```
  count = 1
  while f"input{count}_type" in vals:
      field_type = vals[f"input{count}_type"]    # e.g. "age"
      user_input = vals[f"input{count}"]         # e.g. "35"
      if field_type in reference:
          idx = reference.index(field_type)
          submit_data[idx] = user_input
      count += 1
  ```

- If `reference` is supposed to line up with `"input1"`, `"input2"`, etc., then use the `inputX` index, not the whole key string:

  ```
  count = 1
  while f"input{count}_type" in vals:
      user_input = vals[f"input{count}"]
      idx = count - 1               # or some mapping from count → index
      submit_data[idx] = user_input
      count += 1
  ```

- If `reference` contains semantic names and your form sends both the semantic name and the value, key the lookup on the semantic name instead of `"inputN"`:

  ```
  # Example: vals has {"input1_type": "age", "input1": "35", ...}
  count = 1
  while f"input{count}_type" in vals:
      name = vals[f"input{count}_type"]     # "age"
      val = vals[f"input{count}"]
      if name in reference:
          idx = reference.index(name)
          submit_data[idx] = val
      count += 1
  ```

The core fix is: ensure the string you pass into `reference.index(...)` is actually one of the elements of `reference`, and not a form key like `"input1"` or an unrelated type label.

⁂

1. https://www.geeksforgeeks.org/python/python-key-index-in-dictionary/
2. https://sagarkaurav.hashnode.dev/flask-wtf-forms-dynamic-fields-using-javascript
3. https://discuss.python.org/t/add-safe-get-method-to-list/32555
4. https://python-forum.io/thread-2538.html
5. https://pypi.org/project/WTForms-Dynamic-Fields/
6. https://tedboy.github.io/flask/generated/generated/flask.Request.html
7. https://stackoverflow.com/questions/27824110/flask-wtforms-dynamically-generated-list-of-fields
8. https://community.plotly.com/t/manipulation-of-containers-list-dict-in-dash/64819
9. https://www.youtube.com/watch?v=9MHYHgh4jYc
10. https://www.rmedgar.com/blog/dynamic-fields-flask-wtf/
11. https://www.youtube.com/watch?v=V-y49b95Lv4
12. https://stackoverflow.com/questions/13279399/how-to-obtain-values-of-request-variables-using-python-and-flask
13. https://pythongeeks.org/flask-request-object/
14. https://flask.palletsprojects.com/en/stable/api/
15. https://www.digitalocean.com/community/tutorials/processing-incoming-request-data-in-flask
16. https://www.geeksforgeeks.org/python/using-request-args-for-a-variable-url-in-flask/
17. https://www.browserstack.com/guide/flask-get-request
18. https://www.reddit.com/r/flask/comments/b0ksbs/help_with_dynamically_adding_fields_to_flaskwtf/
19. https://stackoverflow.com/questions/4391697/find-the-index-of-a-dict-within-a-list-by-matching-the-dicts-value
20. https://flask-restplus.readthedocs.io/en/stable/parsing.html

The HTML creating the variables is:

Health Tracker

``` ```

Add Another Input Remove Input

Enter Data

And the receiving function is currently:

```
import sqlite3
import datetime
from bmes_ahmet_loader import *

#=Submit Patient Function#
def submit_patient_data(user_input):
    # Connect to the database
    conn = sqlite3.connect(bmes.userdownloaddir() + "/NWI_DB.db")
    conn.cursor()

#Reference variables for calling database
    reference = ['weight_kg', 'length_cm', 'head_circumference_cm', 'temperature_c',
'heart_rate_bpm',
                'respiratory_rate_bpm', 'oxygen_saturation', 'feeding_frequency_per_day',
                'urination_count_per_day', 'stool_count_per_day']
    reference_types=[float,float,float,float,int,int,int,int,int,int] #Corresponds to each reference for
the kind of variable it is
    vals=[user_input[0]] #Gets the patient id which will never be a null value
    possible_null=['',None] #An array of the posible null values that could trip up sql
```

```
#=Baby Normal Values==========#
    temp_range=[37,38]; #Infant Temperature (c) Range: https://health.clevelandclinic.org/pediatric-vital-signs
    hr_range=[110,160]; #Infant Heart Rate (bpm) Range: https://health.clevelandclinic.org/pediatric-vital-signs
    rr_range=[30,60]; #Infant Respiratory Rate (bpm) Range: https://health.clevelandclinic.org/pediatric-vital-signs
    o2_range=[88,92]; #Infant Oxygen Saturation Range: https://jamanetwork.com/journals/jamapediatrics/article-abstract/515871
    ffr_range=[6,12]; #Infant Feeding Frequency (Per Day) Range: https://www.parents.com/baby/feeding/baby-feeding-chart-how-much-and-when-to-feed-infants-the-first-year/
    u_range=[6,10]; #Infant Urination Frequency (Per Day) Range: https://www.parents.com/breastfeeding-and-wet-diapers-whats-normal-431621#:~:text=Urination%20After%20the%20First%20Week,to%20three%20hours%2C%20is%20fine.
    st_range=[1,8]; #Infant Stool Freguency (Per Day) Range: https://www.childrenscolorado.org/just-ask-childrens/articles/baby-poop-guide/
    normal_ranges=
[tuple(temp_range),tuple(hr_range),tuple(rr_range),tuple(o2_range),tuple(ffr_range),tuple(u_range),tuple(st_range)]; #Infant Temperature (c) Range: https://health.clevelandclinic.org/pediatric-vital-signs
    warnings=[]

for i in range(len(reference)): #Don't need to check the first 1 cause that is the patient ID and, again, not going to be null
        if user_input[i+1] in possible_null: #Checks to see if it is empty
            vals.append(None) #Will make sure that the null value is properly added
        else:
            vals.append(reference_typesi)
            if i >2:
                if vals[i+1]<normal_ranges[i-3][0] or normal_ranges[i-3][1]<vals[i+1]: #Will run this for everything after
                    warnings=[]

return [vals[0]]
```

TBA: data=conn.execute(f"SELECT * FROM patients WHERE patients.id={vals[0]};").fetchone()

The data never arrives in `submit_patient_data` in the shape that function expects, and your AJAX call is currently not sending the dynamic `inputN` fields at all.[1]

Mismatched expectations

- The form generates fields like `input1_type=weight_kg`, `input1=70`, `input2_type=temperature_c`, `input2=37`, etc.[2]
- `submit_patient_data` expects a list `user_input` of length 12: `user_input[^0] = patient_id`, and `user_input[1:]` are ordered in the same order as `reference` (`weight_kg`, `length_cm`, ...).[2]
- Your Flask route code (from the first message) was trying to build that list from `vals` using `reference.index(...)`, but `reference` is not aligned with the keys `"input1"`, `"input1_type"`, etc., so indexing fails or fills the wrong positions.[3]
- In the jQuery `$.post("/submit_health", ...)` call you are only sending `patient_id` and `date`, not the `inputN` fields, so the Flask route receives no measurement values at all.[1]

So both ends are inconsistent: the browser does not send the values, and the server-side mapping from `inputN`/`inputN_type` to the `reference` order is wrong.

Fix the AJAX to send all form fields

Change the `$.post` call to serialize the entire form instead of manually specifying just two fields.[1]

```
$(document).on("submit", "#healthtrackerform", function(e) {
    e.preventDefault();

    // Serialize the form and send all fields (patient_id, inputs, types, date)
    const formData = $(this).serialize();

    $.post("/submit_health", formData, function(data) {
        // existing success handler...
    });
});
```

Make sure your form actually includes `patient_id` and the correct `date` field (you currently post `date_comm` but the form's hidden input is named `date`).[4]
For example, in the HTML set:

```
<input type="hidden" name="patient_id" value="{{ user }}">
<input type="hidden" name="date_comm" id="date_comm" value="">
```

and update the date into `#date_comm` instead of `#date` (or make the names consistent).[4]

Build `user_input` correctly in Flask

In your Flask route for `/submit_health`, reconstruct the ordered `user_input` list from `request.form` using the `inputN_type` values to map into `reference`:

```
from flask import request

@app.route("/submit_health", methods=["POST"])
```

```
def submit_health():
    patient_id = request.form.get("patient_id")
    date = request.form.get("date_comm")  # or "date" if you keep that name

    reference = [
        'weight_kg', 'length_cm', 'head_circumference_cm', 'temperature_c',
        'heart_rate_bpm', 'respiratory_rate_bpm', 'oxygen_saturation',
        'feeding_frequency_per_day', 'urination_count_per_day',
        'stool_count_per_day'
    ]

    # Initialize list aligned with submit_patient_data expectation:
    # [patient_id, weight_kg, length_cm, ...]
    user_input = [patient_id] + [None] * len(reference)

    # Walk through all possible inputN fields
    count = 1
    while f"input{count}_type" in request.form:
        var_name = request.form.get(f"input{count}__type")   # e.g. "weight_kg"
        value = request.form.get(f"input{count}")            # e.g. "70"
        if var_name in reference and value not in ("", None):
            idx = reference.index(var_name)                  # 0-based inside reference
            user_input[idx + 1] = value                      # +1 to skip patient_id
        count += 1

    warnings = submit_patient_data(user_input)
    return jsonify({"message": "Data saved", "warnings": warnings})
```

This makes `user_input[i+1]` line up with `reference[i]`, exactly as `submit_patient_data` expects.[2]

Tweak `submit_patient_data` return and warnings

Inside `submit_patient_data`, you currently reset `warnings` to `[]` inside the loop; that clears any previous warning each time a new abnormal value is found, instead of appending to a list.[2] Change that part to:

```
warnings = []

for i in range(len(reference)):
    if user_input[i+1] in possible_null:
        vals.append(None)
    else:
        vals.append(reference_types[i](user_input[i+1]))
        if i > 2:
            if vals[i+1] < normal_ranges[i-3][^0] or vals[i+1] > normal_ranges[i-3][^1]:
                warnings.append(reference[i])
```

and return something meaningful:

```
return warnings
```

so the AJAX handler can display them.

With these changes:

- The browser sends all dynamic `inputN` and `inputN_type` fields.
- The Flask route maps them into the `reference` order.
- `submit_patient_data` processes the aligned list correctly and can generate warnings.

⁂

1. https://flask.palletsprojects.com/en/stable/patterns/javascript/
2. https://testdriven.io/courses/learn-flask/forms/
3. https://www.geeksforgeeks.org/python/python-key-index-in-dictionary/
4. https://stackoverflow.com/questions/19805763/getting-form-data-with-flask
5. https://www.reddit.com/r/flask/comments/z90tdw/is_it_possible_to_post_html_form_values_to_python/
6. https://stackoverflow.com/questions/17432453/submitting-a-dynamic-form-in-jquery
7. https://laravel.io/forum/05-11-2014-dynamically-created-input-fields-not-present-in-post
8. https://realpython.com/python-serialize-data/
9. https://www.youtube.com/watch?v=dqQDiOzxmwQ
10. https://stackoverflow.com/questions/40490638/jquery-adding-fields-to-a-form-dynamically-post-is-missing-data-when-you-sub
11. https://www.youtube.com/watch?v=wurKVD9haV8
12. https://forum.jquery.com/portal/en/community/topic/best-solution-for-a-dynamic-submit-form
13. https://gist.github.com/giacomomarchioro/f6f3d80d7d48bf8bc8e92486966b7ef3
14. https://www.youtube.com/watch?v=XYx5slbU8B4
15. https://forum.jquery.com/portal/en/community/topic/how-to-get-value-of-dynamically-added-fields
16. https://www.facebook.com/groups/ThaiPGAssociateSociety/posts/1130457767165491/
17. https://stackoverflow.com/questions/12277933/send-data-from-a-textbox-into-flask
18. https://www.youtube.com/watch?v=ap2vxzAZVlg
19. https://pypi.org/project/flask-serialize/
20. https://coderanch.com/t/529988/languages/Submit-dynamically-created-form-jQuery
21. https://www.reddit.com/r/flask/comments/b0ksbs/help_with_dynamically_adding_fields_to_flaskwtf/

Things appear to be working now, but the database is not being saved to correctly:

```
for i in range(len(reference)): #Don't need to check the first 1 cause that is the patient ID and, again, not going to be null
    if user_input[i+1] in possible_null: #Checks to see if it is empty
      vals.append(None) #Will make sure that the null value is properly added
    else:
      vals.append(reference_typesi)
      if i >2:
        if vals[i+1]<normal_ranges[i-3][0] or normal_ranges[i-3][1]<vals[i+1]: #Will run this for everything after
          warnings.append(reference[i])

data=conn.execute(f"SELECT * FROM patients WHERE patients.id={vals[0]};").fetchone()

#Gets the new age in days
  birthdate=datetime.date.fromisoformat(data[5]) #Takes the birth date and turns from text into a format a date object
  today=datetime.date.today() #Gets todays date
  age_days=(today-birthdate).days #Gets the age in days of the patient

#Update Patient Data#
  # conn.execute("INSERT OR IGNORE INTO patient_visit_data (weight_kg) VALUES (1)")
  conn.execute("""INSERT OR IGNORE INTO patient_visit_data (
          id_patient,
          date,
          age_days,
          weight_kg,
          length_cm,
          head_circumference_cm,
          temperature_c,
          heart_rate_bpm,
          respiratory_rate_bpm,
          oxygen_saturation,
          feeding_frequency_per_day,
          urination_count_per_day,
          stool_count_per_day
          ) VALUES (
          ?,?,?,?,?,?,?,?,?,?,?,?,?,?
```

```
                );""",[vals[0],today.isoformat(),age_days]+vals[1:])
    conn.commit()
    conn.close()
    return warnings
```

The insert is very likely being ignored due to constraints, or the values list does not line up with the table schema, so the row never appears even though no error is raised.[1] [2]

Check value ordering and types

`vals` is built as:

- `vals[^0] = patient_id`
- In the loop, you append 10 entries in the order of `reference` (`weight_kg`, `length_cm`, ..., `stool_count_per_day`).

So after the loop, `vals` has length 11:

- `vals[^0]` = id_patient
- `vals[^1]..vals[^10]` = the 10 measurements

In the `INSERT` you pass:

```
[vals[^0], today.isoformat(), age_days] + vals[1:]
```

That produces 13 parameters:

1. id_patient → vals
2. date → today.isoformat()
3. age_days → age_days
4. weight_kg → vals[3]
5. length_cm → vals[4]
6. head_circumference_cm → vals[5]
7. temperature_c → vals[6]
8. heart_rate_bpm → vals[7]
9. respiratory_rate_bpm → vals[8]
10. oxygen_saturation → vals[9]
11. feeding_frequency_per_day → vals[10]
12. urination_count_per_day → vals[11]
13. stool_count_per_day → vals[12]

Verify that:

- `patient_visit_data` has exactly these 13 columns in this order.

- Any `NOT NULL` columns are not getting `None` when you leave a field blank; `INSERT OR IGNORE` will silently skip the insert if a `NOT NULL` or `UNIQUE` constraint is violated, not just primary-key conflicts.[2] [1]

If some of these columns are `NOT NULL` but you allow blanks and convert them to `None`, the whole `INSERT OR IGNORE` will do nothing.

How to confirm what’s wrong

Temporarily remove `OR IGNORE` so SQLite will raise an error instead of discarding the row:

```
conn.execute("""
    INSERT INTO patient_visit_data (
        id_patient,
        date,
        age_days,
        weight_kg,
        length_cm,
        head_circumference_cm,
        temperature_c,
        heart_rate_bpm,
        respiratory_rate_bpm,
        oxygen_saturation,
        feeding_frequency_per_day,
        urination_count_per_day,
        stool_count_per_day
    ) VALUES (
        ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?
    );
""", [vals[^0], today.isoformat(), age_days] + vals[1:])
```

Run a request and see the exception; it will tell you if the parameter count is wrong or a constraint is violated.[13] [14]

Common concrete fixes

- Ensure all 13 columns exist and in the exact order you list them; if the table definition changed, adjust the `INSERT` column list and parameters to match.[15]
- Relax `NOT NULL` constraints for optional measurement columns, or avoid including those columns in the insert when you have `None`.
- If you truly only want to ignore duplicates on a specific unique key (e.g., `(id_patient, date)`), prefer an explicit `ON CONFLICT(id_patient, date) DO NOTHING` instead of `INSERT OR IGNORE`, which hides any constraint problem, not just duplicates.[16] [17]

⁂

1. https://hoelz.ro/blog/with-sqlite-insert-or-ignore-is-often-not-what-you-want
2. https://database.guide/how-on-conflict-works-in-sqlite/
3. https://stackoverflow.com/questions/13279399/how-to-obtain-values-of-request-variables-using-python-and-flask

4. https://flask.palletsprojects.com/en/stable/api/
5. https://www.digitalocean.com/community/tutorials/processing-incoming-request-data-in-flask
6. https://www.geeksforgeeks.org/python/using-request-args-for-a-variable-url-in-flask/
7. https://www.browserstack.com/guide/flask-get-request
8. https://www.reddit.com/r/flask/comments/b0ksbs/help_with_dynamically_adding_fields_to_flaskwtf/
9. https://stackoverflow.com/questions/4391697/find-the-index-of-a-dict-within-a-list-by-matching-the-dicts-value
10. https://flask-restplus.readthedocs.io/en/stable/parsing.html
11. https://sagarkaurav.hashnode.dev/flask-wtf-forms-dynamic-fields-using-javascript
12. https://discuss.python.org/t/add-safe-get-method-to-list/32555
13. https://docs.python.org/3/library/sqlite3.html
14. https://www.sqlitetutorial.net/sqlite-python/insert/
15. https://www.digitalocean.com/community/tutorials/how-to-use-an-sqlite-database-in-a-flask-application
16. https://sqldocs.org/sqlite-upsert-insert-on-conflict/
17. https://sqlite.org/forum/forumpost/703f3867a1e60c50?t=c&unf
18. https://stackoverflow.com/questions/12105198/sqlite-how-to-get-insert-or-ignore-to-work
19. https://www.reddit.com/r/sqlite/comments/132lfz2/insert_or_replace_or_insert_or_ignore_still/
20. https://docs.python.org/it/3.8/library/sqlite3.html
21. https://importpython.com/fixing-length-of-values-does-not-match-length-of-index/
22. https://github.com/sqlalchemy/sqlalchemy/discussions/9675
23. https://writeonly.wordpress.com/2009/02/11/using-a-list-as-a-named-placeholder-in-python-sqlite/
24. https://codedamn.com/news/python/how-to-fix-error-length-of-values-does-not-match-length-of-index
25. https://stackoverflow.com/questions/78488469/sqlite-insert-or-replace-and-on-conflict-do-nothing
26. https://www.daniweb.com/programming/software-development/threads/209685/sqlite-and-placeholder
27. https://xania.org/200711/python-sqlite-gotcha
28. https://stackoverflow.com/questions/63689464/why-does-parameter-count-mismatch
29. https://stackoverflow.com/questions/7991183/how-to-get-variable-length-placeholders-in-a-python-call-to-sqlite3
30. https://sqlite-users.sqlite.narkive.com/yoNlJ8zM/insert-or-ignore-vs-insert-where-not-exists

user_input is initially just "100" (which is the patient id), nothing else. So we are passing the wrong info in from app.py

The Flask route is only putting the patient ID into `user_input`; the measurement fields are never added. You need to build `user_input` from `request.form` using the `inputN_type` → `reference` mapping.

Correct route code to build `user_input`

In `app.py`, your `/submit_health` route should look roughly like this:

```
from flask import request, jsonify

@app.route("/submit_health", methods=["POST"])
def submit_health():
    patient_id = request.form.get("patient_id")

    reference = [
        'weight_kg',
        'length_cm',
        'head_circumference_cm',
        'temperature_c',
        'heart_rate_bpm',
        'respiratory_rate_bpm',
        'oxygen_saturation',
        'feeding_frequency_per_day',
        'urination_count_per_day',
        'stool_count_per_day'
    ]

    # Start with patient_id, then one slot per reference variable
    user_input = [patient_id] + [None] * len(reference)

    count = 1
    while f"input{count}_type" in request.form:
        var_name = request.form.get(f"input{count}_type")   # e.g. "weight_kg"
        value = request.form.get(f"input{count}")           # e.g. "70"
        if var_name in reference and value not in ("", None):
            idx = reference.index(var_name)                 # 0..9
            user_input[idx + 1] = value                     # +1 to skip patient_id
        count += 1

    print("built user_input:", user_input, flush=True)
    warnings = submit_patient_data(user_input)
    return jsonify({"message": "Data saved", "warnings": warnings})
```

Key points:

- `user_input` must have length `1 + len(reference)` before calling `submit_patient_data`.[1]
- The `while f"input{count}_type" in request.form:` loop walks all dynamic fields and places each value at the correct index, based on `reference.index(var_name)`.[1]
- The jQuery submit handler must send the whole form, not just `patient_id`. Use `$(this).serialize()` in your `$.post` call so `input1_type`, `input1`, etc. reach `request.form`.[2] [3]

After this change, prints for `built user_input` should show something like:

```
["100", "70", None, None, "37", ...]
```

and `submit_patient_data` will see all values instead of just `["100"]`.

⁂

1. https://stackoverflow.com/questions/17752301/dynamic-form-fields-in-flask-request-form
2. https://flask.palletsprojects.com/en/stable/patterns/javascript/
3. https://stackoverflow.com/questions/19805763/getting-form-data-with-flask