CO PILOT prompts

i have switched the project subject and need to update to sql tables to be (you can delete the old ones):
User Table:

full_name (string)
email (string, unique)
role (string, 'admin' or 'user')
character_name (string)
character_class (string, e.g., 'Fighter', 'Wizard')
level (integer)
coins (integer)
total_steps (integer)
daily_steps (integer)
last_step_date (date string, e.g., 'YYYY-MM-DD')
visited_locations (JSON array of strings/UUIDs, referencing PointOfInterest.id)
has_selected_class (boolean)
class_change_cost (integer)
class_changes_count (integer)
Achievement Table:

title (string)
description (string)
requirement (number)
requirement_type (string, 'daily_steps', 'total_steps', 'locations_visited', 'consecutive_days')
coin_reward (number)
icon (string)
rarity (string, 'common', 'uncommon', 'rare', 'epic', 'legendary')
UserAchievement Table:

achievement_id (string, Foreign Key to Achievement.id)
unlocked_date (datetime)
progress (number)
PointOfInterest Table:

name (string)
description (string)
latitude (number)

longitude (number)
region (string)
steps_required (number)
coin_reward (number)
location_type (string, 'city', 'dungeon', 'temple', etc.)
Equipment Table:

name (string)
description (string)
type (string, 'weapon', 'armor', 'accessory', 'consumable')
rarity (string, 'common', 'uncommon', 'rare', 'epic', 'legendary')
price (number)
sell_price (number)
class_restricted (JSON array of strings, e.g., ['Fighter', 'Wizard'])
is_permanent (boolean)
stat_bonuses (JSON object, e.g., {strength: 5, dexterity: 2, daily_steps_bonus: 100})
UserEquipment Table:

equipment_id (string, Foreign Key to Equipment.id)
quantity (integer)
is_equipped (boolean)
acquired_date (datetime)
RandomEncounter Table:

latitude (number)
longitude (number)
encounter_type (string, 'treasure', 'monster', 'merchant')
is_active (boolean)
expires_at (datetime)
min_level (integer)
max_level (integer)
Monster Table:

name (string)
challenge_rating (string, e.g., '1/4', '2', '15')
min_level (integer)
max_level (integer)
description (string)
gold_reward (integer)
possible_loot (JSON array of strings/UUIDs, referencing Equipment.id)

here are the names of the old tables. adjust the script accordingly

```
tarted executing query at Line 1
Msg 229, Level 14, State 5, Line 36
The REFERENCES permission was denied on the object 'User', database 'master', schema 'dbo'.
Msg 1088, Level 16, State 20, Line 36
Cannot find the object "User" because it does not exist or you do not have permissions.
Msg 1750, Level 16, State 1, Line 36
Could not create constraint or index. See previous errors.
```

im the dba. and still get this error. how do i fix this

```
Started executing query at Line 1
Commands completed successfully.
8:49:00 PM
Started executing query at Line 97
Msg 229, Level 14, State 5, Line 99
The REFERENCES permission was denied on the object 'User', database 'master', schema 'dbo'.
Msg 1088, Level 16, State 20, Line 99
Cannot find the object "User" because it does not exist or you do not have permissions.
Msg 1750, Level 16, State 1, Line 99
Could not create constraint or index. See previous errors.
```

lets do this step by step. lets first delete all the old tables acording to the images i provided you eariler with

```
Msg 3726, Level 16, State 1, Line 2
Could not drop object 'LocationCode' because it is referenced by a FOREIGN KEY constraint.
Msg 3726, Level 16, State 1, Line 2
Could not drop object 'UserTitle' because it is referenced by a FOREIGN KEY constraint.
```

want a script to automatically find and drop all foreign keys now the date base is empty. edit the script to create the new tables we disccused earlier

ok it worked.
now, since the project changed we need to ditch the old backend code we implemented and create a new inferstructure. here are the new functions with the relevant input/output and sql table (also create the function.json for each function):
getUserData (GET)

Input: Query parameters userId (string), userEmail (string) - sent by Base44 proxy from authenticated user.
Output: JSON object representing the current User record.

```
{
"id": "user_id_uuid",
"created_date": "2023-01-01T10:00:00Z",
"updated_date": "2023-10-26T14:30:00Z",
"full_name": "John Doe",
"email": "john.doe@example.com",
"role": "user",
"character_name": "Sir Reginald",
"character_class": "Paladin",
"level": 5,
"coins": 1250,
"total_steps": 25000,
"daily_steps": 1200,
"last_step_date": "2023-10-26",
"visited_locations": ["poi_id_1", "poi_id_2"],
"has_selected_class": true,
"class_change_cost": 500,
"class_changes_count": 0
}
```

SQL Tables: User
updateUserData (POST)

Input: JSON object containing requestingUser (object with id, email) and fields to update for the User table.

```
{
"requestingUser": { "id": "user_id_uuid", "email": "john.doe@example.com" },
"character_name": "Lady Anya",
"coins": 1200,
"total_steps": 25001,
"daily_steps": 1201
}
```

Output: JSON object of the updated User record. (Same structure as getUserData output).

SQL Tables: User

listPointsOfInterest (GET)

Input: None.

Output: JSON array of PointOfInterest objects.

```
[
{
"id": "poi_id_1",
"created_date": "2023-01-01T10:00:00Z",
"updated_date": "2023-01-01T10:00:00Z",
"created_by": "admin@example.com",
"name": "Silvermoon City",
"description": "A bustling elven city.",
"latitude": 34.0522,
"longitude": -118.2437,
"region": "Elvenwoods",
"steps_required": 5000,
"coin_reward": 50,
"location_type": "city"
}
]
```

SQL Tables: PointOfInterest

listRandomEncounters (GET)

Input: None.

Output: JSON array of RandomEncounter objects.

```
[
{
"id": "encounter_id_1",
"created_date": "2023-10-26T14:00:00Z",
"updated_date": "2023-10-26T14:00:00Z",
"created_by": "system",
"latitude": 34.055,
"longitude": -118.245,
"encounter_type": "monster",
"is_active": true,
"expires_at": "2023-10-27T14:00:00Z",
"min_level": 1,
"max_level": 10
}
]
```

SQL Tables: RandomEncounter

createRandomEncounter (POST)

Input: JSON object containing requestingUser and fields to create a RandomEncounter record.

```
{
"requestingUser": { "id": "user_id_uuid", "email": "john.doe@example.com" },
"latitude": 34.060,
"longitude": -118.250,
"encounter_type": "treasure",
"is_active": true,
"expires_at": "2023-10-27T15:00:00Z"
}
```

Output: JSON object of the newly created RandomEncounter record. (Same structure as listRandomEncounters output).

SQL Tables: RandomEncounter

deleteRandomEncounter (DELETE)

Input: JSON object containing requestingUser and the id of the RandomEncounter to delete.

```
{
"requestingUser": { "id": "user_id_uuid", "email": "john.doe@example.com" },
"id": "encounter_id_1"
}
```

Output: Success confirmation JSON object.

```
{ "success": true, "message": "Random encounter deleted." }
```

SQL Tables: RandomEncounter

listMonsters (GET)

Input: None.

Output: JSON array of Monster objects.

```
[
{
"id": "monster_id_1",
"created_date": "2023-01-01T10:00:00Z",
"updated_date": "2023-01-01T10:00:00Z",
"created_by": "admin@example.com",
"name": "Goblin",
"challenge_rating": "1/4",
"min_level": 1,
"max_level": 5,
"description": "A small, green-skinned menace.",
"gold_reward": 10,
"possible_loot": ["equipment_id_1"]
}
]
```

SQL Tables: Monster

listAchievements (GET)

Input: None.

Output: JSON array of Achievement objects.

```
[
{
"id": "achievement_id_1",
"created_date": "2023-01-01T10:00:00Z",
"updated_date": "2023-01-01T10:00:00Z",
"created_by": "admin@example.com",
"title": "First Step",
"description": "Take 1000 total steps.",
"requirement": 1000,
"requirement_type": "total_steps",
"coin_reward": 25,
"icon": "footprints",
"rarity": "common"
}
]
```

SQL Tables: Achievement

listUserAchievements (GET)

Input: Query parameters userId (string), userEmail (string) - sent by Base44 proxy from authenticated user.
Output: JSON array of UserAchievement objects belonging to the requesting user.

```
[
{
"id": "user_ach_id_1",
"created_date": "2023-01-05T10:00:00Z",
"updated_date": "2023-01-05T10:00:00Z",
"created_by": "john.doe@example.com",
"achievement_id": "achievement_id_1",
"unlocked_date": "2023-01-05T10:00:00Z",
"progress": 1000
}
]
```

SQL Tables: UserAchievement

listEquipment (GET)

Input: None.
Output: JSON array of Equipment objects.

```
[
{
"id": "equipment_id_1",
"created_date": "2023-01-01T10:00:00Z",
"updated_date": "2023-01-01T10:00:00Z",
"created_by": "admin@example.com",
"name": "Basic Sword",
"description": "A simple blade for a new adventurer.",
"type": "weapon",
"rarity": "common",
"price": 100,
"sell_price": 50,
"class_restricted": [],
"is_permanent": true,
"stat_bonuses": {"strength": 2}
}
]
```

SQL Tables: Equipment
listUserEquipment (GET)

Input: Query parameters userId (string), userEmail (string) - sent by Base44 proxy from authenticated user.
Output: JSON array of UserEquipment objects belonging to the requesting user.

```
[
{
"id": "user_equip_id_1",
"created_date": "2023-01-05T10:00:00Z",
"updated_date": "2023-01-05T10:00:00Z",
"created_by": "john.doe@example.com",
"equipment_id": "equipment_id_1",
"quantity": 1,
"is_equipped": false,
"acquired_date": "2023-01-05T10:00:00Z"
}
]
```

SQL Tables: UserEquipment
updateUserEquipment (POST)

Input: JSON object containing requestingUser, the id of the UserEquipment record to update, and fields to modify.

```
{
"requestingUser": { "id": "user_id_uuid", "email": "john.doe@example.com" },
"id": "user_equip_id_1",
"quantity": 2,
"is_equipped": true
}
```

Output: JSON object of the updated UserEquipment record. (Same structure as listUserEquipment output).
SQL Tables: UserEquipment
createUserEquipment (POST)

Input: JSON object containing requestingUser and fields to create a UserEquipment record.

```
{
"requestingUser": { "id": "user_id_uuid", "email": "john.doe@example.com" },
"equipment_id": "equipment_id_2",
"quantity": 1,
"is_equipped": false,
"acquired_date": "2023-10-26T15:00:00Z"
}
```
Output: JSON object of the newly created UserEquipment record. (Same structure as listUserEquipment output).
SQL Tables: UserEquipment
deleteUserEquipment (DELETE)

Input: JSON object containing requestingUser and the id of the UserEquipment record to delete.
```
{
"requestingUser": { "id": "user_id_uuid", "email": "john.doe@example.com" },
"id": "user_equip_id_1"
}
```
Output: Success confirmation JSON object.
```
{ "success": true, "message": "User equipment deleted." }
```
SQL Tables: UserEquipment

create the folders and their relevant code. start with the first 7 functions.

do the same with the last 7 fucntions

create an sql query ro retrive the contens of the table User

i want to make get_user_data dual purpose, i want it to fetch the data by the id and email provided, and if and entry does not exist, i dont want and error but rather an entry to be created with these credentials, and defualt values for the other values in the 'user' table:
role (string, 'admin' or 'user') - user
character_name (string) - adventurer
character_class (string, e.g., 'Fighter', 'Wizard') - fighter
level (integer) - 1
coins (integer) - 0
total_steps (integer) - 0
daily_steps (integer) - 0
last_step_date (date string, e.g., 'YYYY-MM-DD') - that day's date
visited_locations (JSON array of strings/UUIDs, referencing PointOfInterest.id) - empty
has_selected_class (boolean) - false

class_change_cost (integer) - 0
class_changes_count (integer) – 0

you wrote it weirdly, go ovet the implementation again

will this work though? since the function lable is 'GET'

when i test the get user data i get:

```
Error: ('42000', '[42000] [Microsoft][ODBC Driver 18 for SQL Server][SQL
Server]Conversion failed when converting from a character string to
uniqueidentifier. (8169) (SQLExecDirectW)')
```

i give as query email and userId where email is to be unique

i tried the uuid you provided and got instead:

```
Error: ('23000', "[23000] [Microsoft][ODBC Driver 18 for SQL Server][SQL
Server]Cannot insert the value NULL into column 'full_name', table
'zooData.dbo.User'; column does not allow nulls. INSERT fails. (515)
(SQLExecDirectW)")
```

i want to delete the first entry in the user table then change the user id column from uuid to nvarchar(255)

```
9:40:12 PM
Started executing query at Line 1
(0 rows affected)
Msg 5074, Level 16, State 1, Line 6
The object 'DF__User__id__3E1D39E1' is dependent on column 'id'.
Msg 5074, Level 16, State 1, Line 6
The object 'PK__User__3213E83F3E4AFF58' is dependent on column 'id'.
Msg 5074, Level 16, State 1, Line 6
The object 'FK_UserEquipment_User' is dependent on column 'id'.
Msg 5074, Level 16, State 1, Line 6
The object 'FK_UserAchievement_User' is dependent on column 'id'.
Msg 4922, Level 16, State 9, Line 6
ALTER TABLE ALTER COLUMN id failed because one or more objects access this
column.
```

```
9:40:59 PM
Started executing query at Line 1
Msg 3728, Level 16, State 1, Line 6
```

```
'DF__User__id__3E1D39E1' is not a constraint.
Msg 3727, Level 16, State 0, Line 6
Could not drop constraint. See previous errors.
```

i want to drop all constraints on the id columnautomatically

```
{
"error": "Invalid JSON response from Azure",
"azureStatus": 500,
"azureBody": "Error: Object of type date is not JSON serializable"
}
```

i get this when using the get user data

```
{
"error": "Invalid JSON response from Azure",
"azureStatus": 500,
"azureBody": "Error: ('HYT00', '[HYT00] [Microsoft][ODBC Driver 18 for SQL Server]Login timeout expired (0) (SQLDriverConnect)')"
}
```

what does this mean?

regarding this line, both user id and achievement id are called 'id' in thier respective sql tables. how to fix that

but the column names are already set

i get this when testing listUserAchievements:

```
{
"error": "Failed to proxy request to Azure",
"details": "Unexpected token 'M', "Missing user_id" is not valid JSON"
}
```

this is the output i got from the base44 agent:
That's a great question, and it helps clarify the responsibility!

You're seeing "Missing user_id" because:

Yes, the listUserAchievements function should expect a user ID. This function is intended to retrieve achievements specific to the logged-in user, so it needs to know which user's achievements to list.
The Base44 backend function (functions/listUserAchievements.js) is correctly passing the user ID. We ensure that the userId (and userEmail) are appended as

query parameters to the URL when calling your Azure Function.
Therefore, the problem is most likely on your Azure Function's side.

Your Azure Function for listUserAchievements is probably expecting the user ID parameter to be named user_id (with an underscore) or is looking for it in a different part of the request (e.g., in the request body) rather than the URL query parameters.

What you need to check in your Azure Function:

Parameter Name: Look at how your Azure Function code tries to extract the user_id. It should be looking for userId (camelCase) from the query parameters.
For example, in Python Azure Functions, if you're using req.params, it might be req.params.get('userId'). If using req.get_query_parameters(), it would be req.get_query_parameters().get('userId').
Error Handling: Ensure that if userId is not found, your Azure Function returns a proper JSON error object (e.g., {"error": "Missing userId"}) instead of a plain string. This helps the Base44 proxy understand the response.
Once you adjust your Azure Function to correctly read the userId query parameter, this error should be resolved.

now i get

```
{
"error": "('42000', '[42000] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]Conversion failed when converting from a character string to uniqueidentifier. (8169) (SQLExecDirectW)')"
}
```

which probably means i need to change to type of user id in UserAchievements (and userEquipment probably) to NVARCHAR(255)

i want to populate the relevant sql tables with the data here in the csv i provided. one for monsters, one for equipment and one for achievements

i want a Python script to automate this

```
C:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend>C:/Users/omrik/OneDrive/Desktop/IoT_Project/zoo-sync/backend/.venv/Scripts/activate.bat

(.venv) C:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend>C:/Users/omrik/OneDrive/Desktop/IoT_Project/zoo-sync/backend/.venv/Scripts/python.exe c:/Users/omrik/OneDrive/Desktop/IoT_Project/zoo-sync/backend/c.py
Traceback (most recent call last):
```

```
File "c:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend\c.py", line
5, in <module>
conn = pyodbc.connect(conn_str)
^^^^^^^^^^^^^^^^^^^^^^^
pyodbc.InterfaceError: ('IM002', '[IM002] [Microsoft][ODBC Driver Manager] Data
source name not found and no default driver specified (0) (SQLDriverConnect)')
```

```
(.venv) C:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-
sync\backend>C:/Users/omrik/OneDrive/Desktop/IoT_Project/zoo-
sync/backend/.venv/Scripts/python.exe
c:/Users/omrik/OneDrive/Desktop/IoT_Project/zoo-sync/backend/c.py
Traceback (most recent call last):
File "c:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend\c.py", line
43, in <module>
import_equipment(r'c:\Users\omrik\Downloads\Equipment_export.csv')
File "c:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend\c.py", line
12, in import_equipment
cursor.execute("""
pyodbc.ProgrammingError: ('42000', '[42000] [Microsoft][ODBC Driver 18 for SQL
Server][SQL Server]Conversion failed when converting from a character string to
uniqueidentifier. (8169) (SQLExecDirectW)')
```

```
(.venv) C:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend>
```

```
4:16:52 PM
Started executing query at Line 1
Msg 5074, Level 16, State 1, Line 1
The object 'DF__Equipment__id__56E8E7AB' is dependent on column 'id'.
Msg 5074, Level 16, State 1, Line 1
The object 'PK__Equipmen__3213E83FC36EC77B' is dependent on column 'id'.
Msg 5074, Level 16, State 1, Line 1
The object 'FK_UserEquipment_Equipment' is dependent on column 'id'.
Msg 4922, Level 16, State 9, Line 1
ALTER TABLE ALTER COLUMN id failed because one or more objects access this
column.
```

```
Traceback (most recent call last):
File "c:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend\c.py", line
44, in <module>
import_achievement(r'c:\Users\omrik\Downloads\Achievement_export.csv')
```

```
File "c:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend\c.py", line 24, in import_achievement
cursor.execute("""
pyodbc.ProgrammingError: ('42000', '[42000] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]Conversion failed when converting from a character string to uniqueidentifier. (8169) (SQLExecDirectW)')
```

i dont know to actual constraint names

drop them automatically!

```
Traceback (most recent call last):
File "c:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend\c.py", line 43, in <module>
import_equipment(r'c:\Users\omrik\Downloads\Equipment_export.csv')
File "c:\Users\omrik\OneDrive\Desktop\IoT_Project\zoo-sync\backend\c.py", line 12, in import_equipment
cursor.execute("""
pyodbc.IntegrityError: ('23000', "[23000] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]Violation of PRIMARY KEY constraint 'PK_Equipment'. Cannot insert duplicate key in object 'dbo.Equipment'. The duplicate key value is (6890d16ea24803d9f43ab4c5). (2627) (SQLExecDirectW); [23000] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]The statement has been terminated. (3621)")
```

attached are the columns of the userEquipment table. adjust list_user_equipment accordingly

here are the columns of the userEquipment table. adjust list_user_equipment accordingly:
id
user_id
equipment_id
quantity
is_equipped
acquired_date

what is returned from the list_equipment function?

i want to return every column. not just these fields

do the same for list_achievements

This error occurs because item.class_restricted is not an array, so it doesn't have the .join() method. This happens when your Azure Function returns class_restricted as a string or null instead of an array.
For your Azure Function: Make sure your listEquipment function returns class_restricted as a JSON array, not a string. For example:

✅ Good: "class_restricted": ["Fighter", "Paladin"]
❌ Bad: "class_restricted": "Fighter,Paladin" or "class_restricted": null
This fix will prevent the error and make the component more robust against unexpected data formats.

i need to create sevral new functions:

GET /api/inventory (Get User's Full Inventory)
Purpose: To fetch a list of all items a user owns, complete with full details (name, description, stats, etc.). This is different from listUserEquipment because it needs to combine data from both your UserEquipment and Equipment tables.
Input: userId (passed as a query parameter from my proxy function).
Logic:
Get all records from UserEquipment that match the userId.
For each record, use the equipment_id to look up the full item details from the Equipment table.
Return an array of objects, where each object contains the combined data (e.g., userEquipmentId, quantity, is_equipped, name, description, stat_bonuses, etc.).
Example JSON response for one item:

```
{
"user_equipment_id": "user_equip_abc",
"equipment_id": "equip_001",
"quantity": 1,
"is_equipped": true,
"name": "Iron Longsword",
"description": "A basic but reliable longsword.",
"type": "weapon",
"rarity": "common",
"price": 50,
"sell_price": 10,
"stat_bonuses": { "strength": 1 }
}
```

POST /api/equipment/equip (Equip an Item)
Purpose: To mark an item as equipped and unequip any other item in the same slot.
Input: A JSON body containing { "userEquipmentId": "some_id" }. The userId will also be available from the request.
Logic:
Find the UserEquipment record by its ID.
Determine the item's type (e.g., "weapon").
Find any other item of the same type that is currently equipped by the user and set its is_equipped status to false.
Set the is_equipped status of the target userEquipmentId to true.
Response: A success message or the updated item data.

POST /api/equipment/use (Use a Consumable Item)
Purpose: To use a consumable item, which decreases its quantity or removes it from the inventory.
Input: A JSON body containing { "userEquipmentId": "some_id" }.
Logic:
Find the UserEquipment record by its ID.
Check if the item is a consumable.
If quantity > 1, decrement the quantity by 1.
If quantity == 1, delete the UserEquipment record entirely.
Response: A success message confirming the item was used.

POST /api/equipment/unequip (Unequip an Item)
Purpose: To explicitly unequip an item, leaving its slot empty.
Input: A JSON body containing { "userEquipmentId": "some_id" }. The userId is also available.
Logic:
Find the UserEquipment record by its userEquipmentId and userId.
Set its is_equipped status to false.
Response: A success message.

i want a script to clear the point of intresrt table and enter:
-- Example TAU buildings (you'll need to verify/adjust these coordinates)
INSERT INTO PointOfInterest (name, description, latitude, longitude, region, steps_required, coin_reward, location_type) VALUES
('Smolarz Auditorium', 'The main auditorium where major lectures and events take place', 32.1133, 34.8044, 'Central Campus', 100, 25, 'temple'),
('Cymbalista Synagogue', 'The beautiful modern synagogue designed by Mario

```
Botta', 32.1139, 34.8036, 'Central Campus', 200, 30, 'temple'),
('TAU Library', 'The central library with vast knowledge repositories', 32.1129, 34.8039, 'Central Campus', 150, 20, 'city'),
('Exact Sciences Building', 'Home to mathematics, computer science, and physics departments', 32.1135, 34.8048, 'Science Quarter', 300, 40, 'ruins'),
('Engineering Faculty', 'Where future engineers craft their skills', 32.1142, 34.8052, 'Engineering Quarter', 250, 35, 'city'),
('Student Union Building', 'The social hub of campus life', 32.1127, 34.8041, 'Central Campus', 50, 15, 'tavern'),
('Antin Square', 'The main gathering place and heart of the campus', 32.1131, 34.8042, 'Central Campus', 75, 20, 'city'),
('TAU Art Gallery', 'Exhibitions and cultural events venue', 32.1136, 34.8038, 'Cultural Quarter', 400, 50, 'temple');
```

make sure create_random_encounter follows this, if not - make it so:
Input:

Query Parameter: userId (e.g., .../api/random-encounter/create?userId=some_user_id)
Request Body (JSON): An object with the following structure:

```
{
"latitude": 32.113,
"longitude": 34.804,
"encounter_type": "monster", // or "treasure"
"is_active": true,
"expires_at": "2024-08-15T12:00:00.000Z", // ISO 8601 format
"min_level": 1,
"max_level": 20
}
```

Output (on success):

Status Code: 200 or 201
Response Body (JSON): The full RandomEncounter object that was just created in your database, including its new id. This is crucial for the frontend to track the new encounter.

```
{
"id": "newly_created_encounter_id",
"latitude": 32.113,
"longitude": 34.804,
"encounter_type": "monster",
"is_active": true,
"expires_at": "2024-08-15T12:00:00.000Z",
"min_level": 1,
"max_level": 20
}
```

we have the same problem when converting from a character string to uniqueidentifier when trying to submit data to the 'Monster' table. create a query to automaticly drop constraints and change the id to NVARCHAR(255)

i need to create sevral new functions:
Location Tracking & Proximity Functions
/api/player-location/update - POST
Input:

```
{
"latitude": 32.1131,
"longitude": 34.8042,
"timestamp": "2024-01-15T10:30:00Z",
"accuracy": 5.0
}
```

Output:

```
{
"success": true,
"message": "Location updated successfully"
}
```

SQL Table: UserLocation

/api/encounters/nearby - GET
Input: Query params: ?userId={userId}&latitude={lat}&longitude={lng}&radius=10
Output:

```
{
"encounters": [
{
"id": "encounter-123",
"latitude": 32.1132,
"longitude": 34.8043,
"encounter_type": "monster",
"distance_meters": 8.5,
"is_active": true,
"expires_at": "2024-01-16T10:30:00Z"
}
]
}
```

SQL: Uses existing RandomEncounter table

/api/encounters/activate - POST
Input:

```
{
"encounter_id": "encounter-123",
"user_latitude": 32.1131,
"user_longitude": 34.8042
}
```

Output:

```
{
"success": true,
"encounter": {
"id": "encounter-123",
"encounter_type": "monster",
"monster_id": "monster-456",
"verified_proximity": true
}
}
```

another batch of functions to implement:
/api/monsters/{monsterId}/details - GET
Input: Path param: monsterId
Output:

```
{
"id": "monster-456",
"name": "Adult Red Dragon",
"challenge_rating": "17",
"description": "The dragon's scales gleam like molten metal...",
"gold_reward": 500,
"min_level": 15,
"max_level": 20,
"image_url": "https://zoosync2.blob.core.windows.net/reportphoto/Adult Red
Dragon.jpg"
}
```

image_url is constructed
from https://zoosync2.blob.core.windows.net/reportphoto/ + Monster.name + .jpg (or jpeg). SQL: Uses existing Monster table. No ALTER TABLE needed.

/api/monsters/search-by-name - GET
Input: Query param: ?name=Adult Red Dragon Output:

```
{
"monster": {
"id": "monster-456",
"name": "Adult Red Dragon",
"image_url": "https://zoosync2.blob.core.windows.net/reportphoto/Adult Red
Dragon.jpg"
}
}
```

Notes: image_url is constructed
from https://zoosync2.blob.core.windows.net/reportphoto/ + Monster.name + .jpg (or jpeg). SQL: Uses existing Monster table.

next:
/api/player-encounters/create - POST
Input:

```
{
"monster_id": "monster-456",
"encounter_result": "victory", // "victory", "fled", "defeated"
"encounter_date": "2024-01-15T10:30:00Z",
"location_latitude": 32.1131,
"location_longitude": 34.8042
}
```

Output:

```
{
"success": true,
"encounter_id": "player-encounter-789"
}
```

SQL Table: PlayerMonsterEncounter

/api/player-encounters/list - GET

Input: Query param: ?userId={userId} Output:

```
{
"encountered_monsters": [
{
"monster_id": "monster-456",
"name": "Adult Red Dragon",
"first_encountered": "2024-01-15T10:30:00Z",
"total_encounters": 3,
"victories": 2,
"defeats": 0,
"fled": 1
}
]
}
```

SQL: Joins PlayerMonsterEncounter and Monster tables.

/api/compendium/{userId} - GET

Input: Path param: userId Output:

```
{
"compendium": [
{
"monster_id": "monster-456",
"name": "Adult Red Dragon",
"challenge_rating": "17",
"description": "The dragon's scales gleam like molten metal...",
"image_url": "https://zoosync2.blob.core.windows.net/reportphoto/Adult Red Dragon.jpg",
"first_encountered": "2024-01-15T10:30:00Z",
"encounter_stats": {
"total_encounters": 3,
"victories": 2,
"defeats": 0,
"fled": 1
}
}
]
}
```

SQL: Joins PlayerMonsterEncounter and Monster tables.

next:

/api/combat/initiate - POST

Input:

```
{
"encounter_id": "encounter-123",
"monster_id": "monster-456"
}
```

Output:

```
{
"success": true,
"combat_session": {
"id": "combat-999",
"monster": {
"name": "Adult Red Dragon",
"challenge_rating": "17",
"description": "The dragon's scales gleam...",
"gold_reward": 500,
"image_url": "https://zoosync2.blob.core.windows.net/reportphoto/Adult Red Dragon.jpg"
},
"player_hp": 100, // Or whatever player's current HP is
"monster_hp": 100, // Or monster's starting HP
"log": ["Combat initiated with Adult Red Dragon!"]
}
}
```

SQL Table: CombatSession

/api/combat/flee - POST

Input:

```
{
"combat_session_id": "combat-999"
}
```

Output:

```
{
"success": true,
"message": "Successfully fled from combat.",
"outcome": "fled"
}
```

SQL: Updates CombatSession status and calls /api/player-encounters/create to record fled.

/api/combat/victory - POST

Input:

```
{
"combat_session_id": "combat-999"
}
```

Output:

```
{
"success": true,
"message": "Victory achieved!",
"gold_earned": 500,
"xp_earned": 100,
"outcome": "victory"
}
```

SQL: Updates CombatSession status, updates user's gold/XP via User table, and calls /api/player-encounters/create to record victory.

/api/user/location/update - PUT

Input:

```
{
"latitude": 32.1131,
"longitude": 34.8042
}
```

Output:

```
{
"success": true,
"message": "User location updated."
}
```

SQL: Updates UserLocation table (or creates new entry).

/api/user/location/get - GET
Input: Query param: ?userId={userId} Output:

```
{
"latitude": 32.1131,
"longitude": 34.8042,
"timestamp": "2024-01-15T10:30:00Z"
}
```

SQL: Queries UserLocation for the most recent entry for the userId.

i see that some folders miss the **init**.py file
combat flee
combat victory
get_user_location
initiate combat
update user location
go over the history of the conversation where i provided you with the apropriate information to implement these,

this is the error i get for update_player_location

```
{
"success": false,
"message": "('23000', "[23000] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]Cannot insert the value NULL into column 'id', table 'zooData.dbo.UserLocation'; column does not allow nulls. INSERT fails. (515) (SQLExecDirectW)")"
}
```

id is NVARCHAR(255). use the best method.

write the sql query to change the id type in UserLocation to UNIQUEIDENTIFIER NOT NULL. also consider changes to update_user_location if neccesery

Base 44 Prompts – not word by word, history doesn't save.

I want to build a fitness tracking app with a Dungeons & Dragons theme. Let's start with a dashboard to show a user's activity and a map to explore

Create a basic navigation layout for these pages. I'd like a fantasy-inspired design.

Now, let's add a character component. Users should have a character name, class, and level. Display their current gold. Create a dedicated 'Character' page

Implement an inventory system. Users should be able to collect and equip items. Create 'Equipment' and 'UserEquipment' entities, and an 'Inventory' page.

Players need a way to get new gear. Create a 'Shop' page where they can buy equipment for gold and sell their existing items.

Integrate real-time GPS tracking into the map. I want users to see their current location and track steps automatically as they move. Steps should update daily and overall total steps.

My steps aren't updating correctly in the app, and the total steps often appear as zero. Can you investigate the step counting logic and how it syncs with the user's data

Populate the map with 'Points of Interest' (POIs). Users should be able to visit these locations when they are nearby and earn gold as a reward. Create a 'PointOfInterest' entity

When visiting a location on the map, I'm getting an error message like 'Unexpected non-whitespace character after JSON'. It seems the data from the backend isn't being parsed correctly by the frontend after I interact with a POI

Let's make the map more dynamic! Random encounters (monsters, treasure) should appear. When a user is near an encounter, they can interact with it.

The random encounters aren't appearing on the map at all. Can you check why they're not being created or displayed

The encounters are still not showing up. It seems there's an error about 'expires_at' being missing when trying to create them. Also, can we make sure only 3 encounters spawn when the area is clear

The random encounters appear too randomly, or not at all. I want encounters to spawn consistently (e.g., 3 at a time) when the area is clear, and their expiration time should depend on how far away they are

When a user successfully interacts with a monster encounter (e.g., defeats a monster), it should be recorded in a 'Monster Compendium'. Create a new page for this.

Add an 'Achievement' system. Users should earn achievements for milestones like total steps, daily steps, and locations visited. Create a page to display these achievements

I'm still having issues with daily steps. Total steps update fine, but daily steps always reset to zero, and the app sometimes says '0 steps recorded' when I stop GPS. Ensure the daily step count and last step date are passed and processed correctly.

t also seems the GPS context is lost if I navigate away from the map page. Please add a clear warning that users need to stay on the map page and explicitly 'Stop GPS' to save their progress.

"Now that the GPS and steps are working better, let's re-enable the distance requirement for interacting with POIs and encounters (e.g., within 30 meters).

"The accuracy of random encounter spawning is very important. Please ensure encounters only generate if my GPS accuracy is within 25 meters.