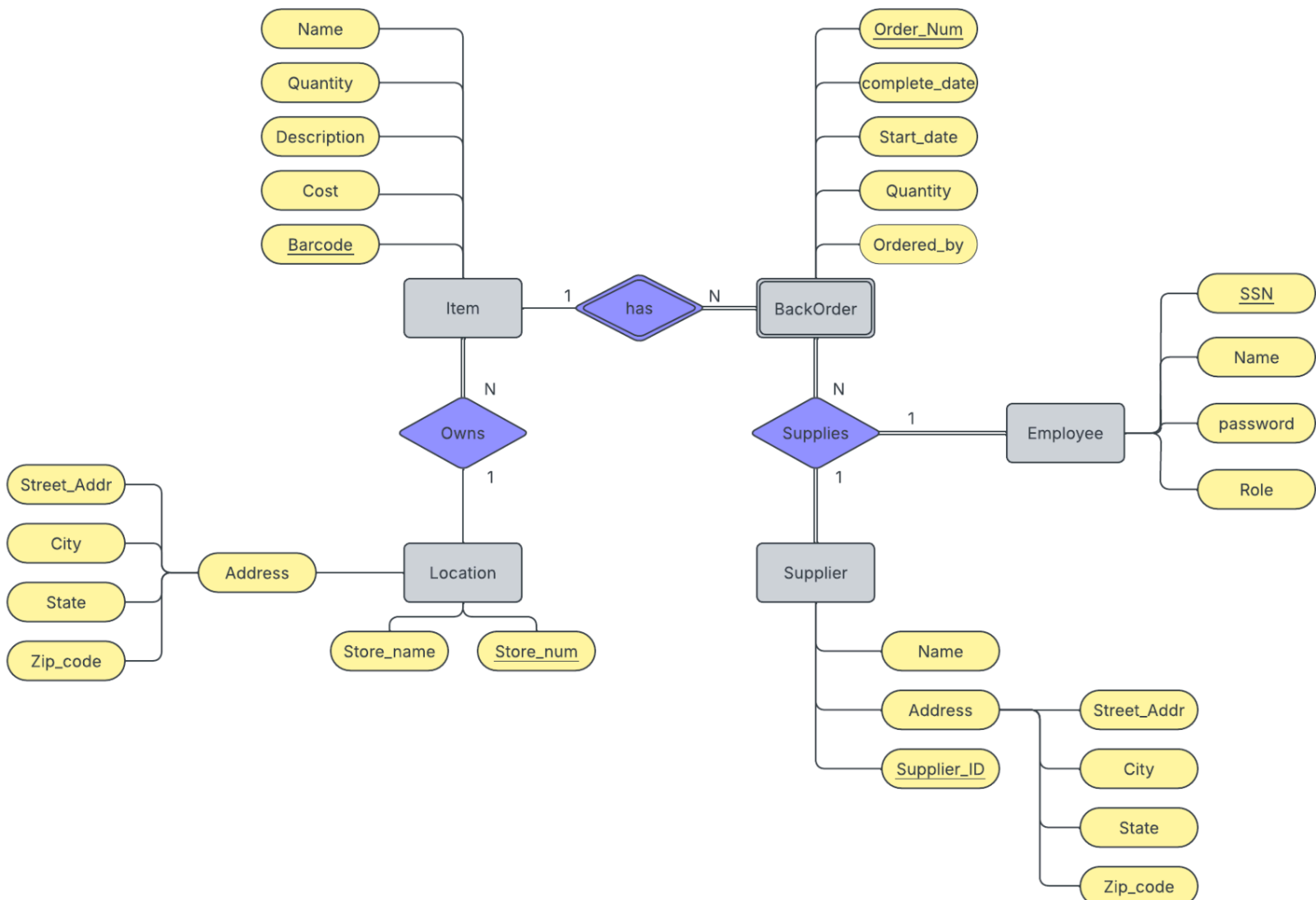# Project Phase III - Inventory Database
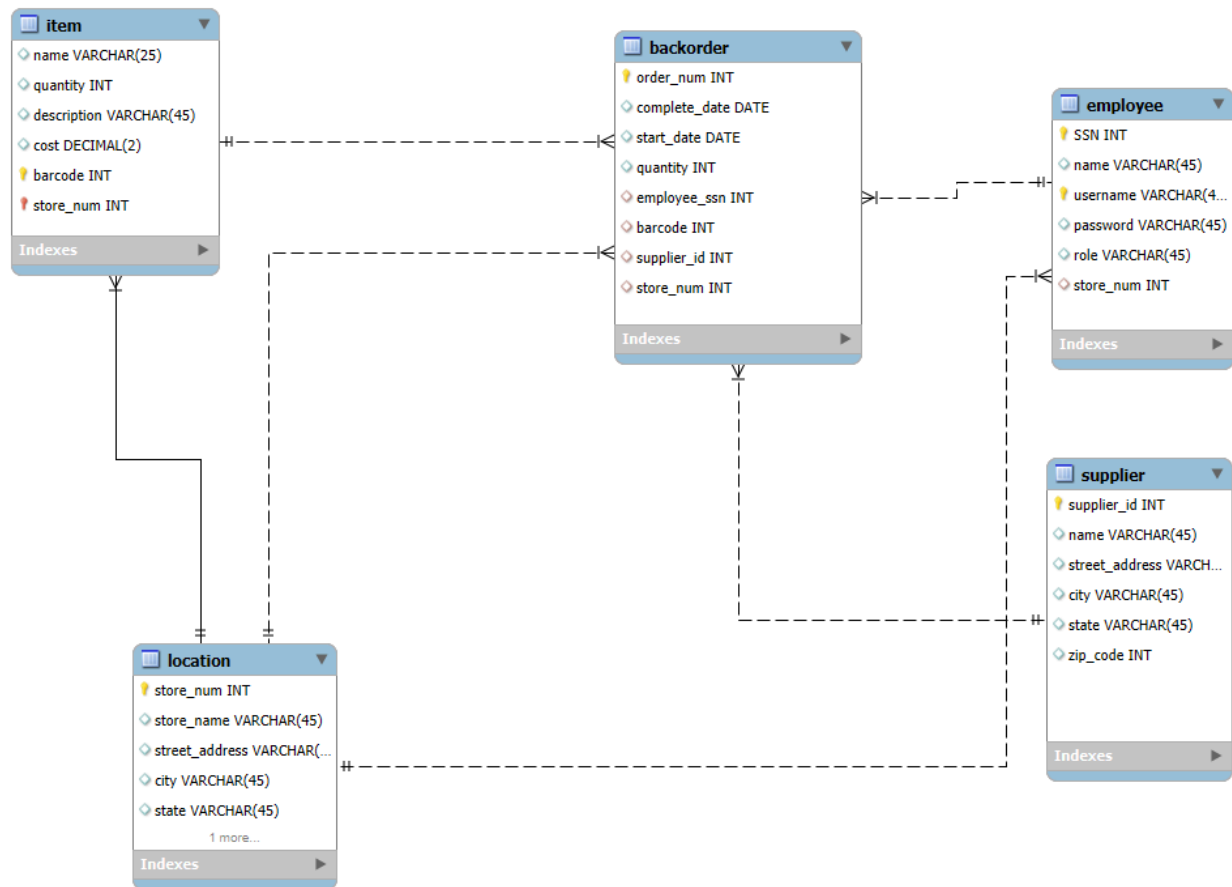*Connor Winning*

- **Problem Statement:**
  My project is a perpetual inventory system that will keep track of the current inventory in a store. Every entity in the database should have a specific bar code that will be used as a key to search for. Items should be able to be added and removed from the inventory system. It must store the cost, name, quantity, and other information of every item in the store, and it must store back-order information about items currently ordered. Employee information should be stored for each location. Other information includes supplier information, supply order logging, and store location information. This will be done through a simple menu-style command-line interface.

- **Conceptual Database Design:**

- Logical Database Design:

item
- name VARCHAR(25)
- quantity INT
- description VARCHAR(45)
- cost DECIMAL(2)
- barcode INT
- store_num INT

Indexes

backorder
- order_num INT
- complete_date DATE
- start_date DATE
- quantity INT
- employee_ssn INT
- barcode INT
- supplier_id INT
- store_num INT

Indexes

employee
- SSN INT
- name VARCHAR(45)
- username VARCHAR(4...
- password VARCHAR(45)
- role VARCHAR(45)
- store_num INT

Indexes

supplier
- supplier_id INT
- name VARCHAR(45)
- street_address VARCH...
- city VARCHAR(45)
- state VARCHAR(45)
- zip_code INT

Indexes

location
- store_num INT
- store_name VARCHAR(45)
- street_address VARCHAR(...
- city VARCHAR(45)
- state VARCHAR(45)
- 1 more...

Indexes

- **Application Program Design:**

Login():
        Username = prompt for username
        Password = prompt for password
        Query employee table
        if(Check for username and password match):
                Is_logged_in = true
        if(employee is manager):
                Is_admin = true

add_item():
        Read JSON data from the front end
        Name = JSON.get(name)
        Quantity = JSON.get(Quantity)
        Description = JSON.get(Description)
        Cost = JSON.get(Cost)
        Barcode = JSON.get(Barcode)
        Store_num = Query employee table for store num by employee username
        Add item data to the item table

create_backorder():
        Order_num = generate_order_num()
        Start_date = get_current_date()
        Employee_username = get username from instance
        Store_num = query employee table for store_num by username
        Read JSON data from the front end
        Complete_date = JSON.get(name)
        Quantity = JSON.get(quantity)
        Barcode = JSON.get(barcode)
        Supplier_id = get selected supplier from front end

remove_item():
        Do delete query in item table

remove_backorder():
        Do delete query in backorder table

Remove_location():
    if(not is_admin):
        Prompt user ("Admin Privlages Required for Action")
        Return
    Do delete query in location table
    Cascade to item table
    Cascade to employee table
    Cascade to backorder table

Add_location():
    If (not is_admin):
        Prompt user ("Admin Privlages Required for Action")
        Return
    Read JSON data from the front end
    Store_name = JSON.get(store_name)
    Address = JSON.get(address)
    Store_num = generate_store_num
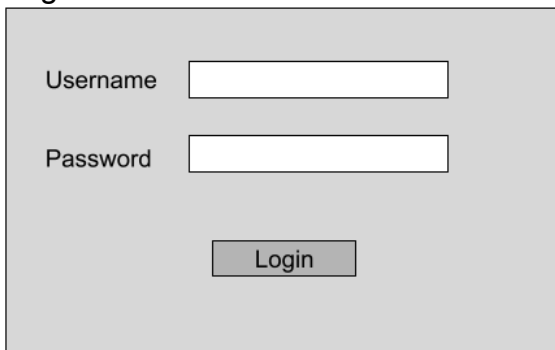
show_items():
    store_num = query store_num from employee table
    Items[] = query filtered data from the item table by store_num
    Show items in the front end

- **Graphical User Interface Design:**

Login Window:

Username	
Password	
	Login

Add Item Window:

Item

Item Name [   ]

Quantity [   ]

Cost [   ]

Description [   ]

Barcode [   ]

Supplier

Suppliers [   ]
(drop down menu)
(select Supplier)

[submit]

Create Backorder Window:

Item

Barcode [   ]

Order Date [   ]

Est. Delivery Date [   ]

Quantity [   ]

Supplier

Suppliers [   ]
(drop down menu)
(select Supplier)

[submit]

Installation Instructions:

1. Prerequisites

Make sure Python 3 or later is installed on your system. You can check with:

```
python -version
```

Also, requires docker to be installed on your system.

2. Navigate to the Project Directory

Open a terminal and navigate to the project root. It should look like this:

```
...\storeInventory\
```

3. Install Dependencies

Run the following command to install all required Python packages:

```
pip install -r requirements.txt
```

4. Start the Database

Before running the application, you need to start the database. Use the following command:

```
python db/start.py

[+] Running 2/2
 ✔ Network storeinventory_default  Created      0.0s
 ✔ Container store_inventory_db    Started      0.3s
* * * * * * * * * * * * * * * * * * * * * Database is UP!
Database schema initialized from setup.sql.
```

5. Run the Application

Once the database is running, you can start the client application:

```
python main.py
```

User Manual

- Default Admin Account

A default admin account is already added for testing:

- **Username:** `admin`

- **Password:** `password`

- Error while running db/start.py

More than likely when an error occurs during this step, it is due to a permission error. Open the repository in a raised cmd terminal or as sudo in bash.

Next, run the command "python db/start.py" again

- Stopping the database

To stop the database run the stop script:

```
Python db/stop.py
```