Below is the source code for a **simplified asset inventory management system** using **HTML, CSS, and JavaScript**, integrated with **Python (Flask)** for backend logic and **PostgreSQL** as the database. The system will allow you to add, delete, modify, and manage assets with CRUD operations and also generate reports on a daily, weekly, monthly, and yearly basis.

**Project Overview:**

1. **HTML** for structure.
2. **CSS** for styling.
3. **JavaScript** for dynamic client-side behavior.
4. **Python (Flask)** for backend logic.
5. **PostgreSQL** for data storage.
6. CRUD (Create, Read, Update, Delete) operations for asset management.
7. Generate reports based on asset inventory.

**1. HTML Structure (index.html)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Asset Inventory Management</title>

<link rel="stylesheet" href="static/style.css">

</head>

<body>

<div class="container">

<h1>Asset Inventory Management System</h1>

<!-- Form for adding/updating asset -->

<form id="assetForm">

<label for="assetId">Asset ID:</label>

<input type="text" id="assetId" name="assetId" required>

<label for="description">Description:</label>

<input type="text" id="description" name="description" required>

<label for="comments">Comments:</label>

<input type="text" id="comments" name="comments">

<label for="assetType">Asset Type:</label>

<input type="text" id="assetType" name="assetType" required>

<button type="submit">Add/Update Asset</button>

</form>

<!-- Display Assets -->

<div id="assetList"></div>

<!-- Button to generate reports -->

<button id="generateReports">Generate Reports</button>

<div id="reportResults"></div>

</div>

<script src="static/app.js"></script>

</body>

</html>

**2. CSS Styling (static/style.css)**

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

}

.container {

max-width: 800px;

margin: 50px auto;

padding: 20px;

background-color: white;

border-radius: 10px;

box-shadow: 0 0 15px rgba(0, 0, 0, 0.2);

}

h1 {

text-align: center;

color: #333;

}

label {

display: block;

margin: 15px 0 5px;

}

input {

width: 100%;

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

padding: 10px 20px;

background-color: #28a745;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #218838;

}

#assetList, #reportResults {

margin-top: 20px;

}

**3. JavaScript for Client-Side Logic (static/app.js)**

document.getElementById('assetForm').addEventListener('submit', submitForm);

document.getElementById('generateReports').addEventListener('click', generateReports);

function submitForm(e) {

e.preventDefault();

const assetData = {

assetId: document.getElementById('assetId').value,

description: document.getElementById('description').value,

comments: document.getElementById('comments').value,

assetType: document.getElementById('assetType').value

};

fetch('/add\_asset', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(assetData)

})

.then(res => res.json())

.then(data => {

alert(data.message);

document.getElementById('assetForm').reset();

fetchAssets();

})

.catch(err => console.log('Error:', err));

}

function fetchAssets() {

fetch('/assets')

.then(res => res.json())

.then(data => {

const assetList = document.getElementById('assetList');

assetList.innerHTML = '<h3>Assets List:</h3>';

data.forEach(asset => {

assetList.innerHTML += `<p>${asset.assetId}: ${asset.description} (${asset.assetType})</p>`;

});

})

.catch(err => console.log('Error:', err));

}

function generateReports() {

fetch('/reports')

.then(res => res.json())

.then(data => {

const reportResults = document.getElementById('reportResults');

reportResults.innerHTML = '<h3>Generated Reports:</h3>';

data.forEach(report => {

reportResults.innerHTML += `<p>${report}</p>`;

});

})

.catch(err => console.log('Error:', err));

}

// Load assets on page load

fetchAssets();

**4. Python Flask Backend (app.py)**

from flask import Flask, request, jsonify, render\_template

import psycopg2

app = Flask(\_\_name\_\_)

# Database connection

def get\_db\_connection():

conn = psycopg2.connect(

dbname='asset\_inventory',

user='your\_username',

password='your\_password',

host='localhost',

port='5432'

)

return conn

# Initialize database table if not exists

def create\_asset\_table():

conn = get\_db\_connection()

cur = conn.cursor()

cur.execute('''

CREATE TABLE IF NOT EXISTS assets (

id SERIAL PRIMARY KEY,

asset\_id VARCHAR(50) NOT NULL,

description TEXT NOT NULL,

comments TEXT,

asset\_type TEXT NOT NULL

);

''')

conn.commit()

cur.close()

conn.close()

@app.route('/')

def index():

return render\_template('index.html')

# Add or update asset

@app.route('/add\_asset', methods=['POST'])

def add\_asset():

data = request.get\_json()

asset\_id = data['assetId']

description = data['description']

comments = data['comments']

asset\_type = data['assetType']

conn = get\_db\_connection()

cur = conn.cursor()

cur.execute('SELECT \* FROM assets WHERE asset\_id = %s', (asset\_id,))

asset = cur.fetchone()

if asset:

# Update existing asset

cur.execute('''

UPDATE assets SET description = %s, comments = %s, asset\_type = %s WHERE asset\_id = %s

''', (description, comments, asset\_type, asset\_id))

else:

# Insert new asset

cur.execute('''

INSERT INTO assets (asset\_id, description, comments, asset\_type)

VALUES (%s, %s, %s, %s)

''', (asset\_id, description, comments, asset\_type))

conn.commit()

cur.close()

conn.close()

return jsonify({"message": "Asset added/updated successfully"})

# Get all assets

@app.route('/assets', methods=['GET'])

def get\_assets():

conn = get\_db\_connection()

cur = conn.cursor()

cur.execute('SELECT \* FROM assets')

assets = cur.fetchall()

asset\_list = []

for asset in assets:

asset\_list.append({

"assetId": asset[1],

"description": asset[2],

"comments": asset[3],

"assetType": asset[4]

})

cur.close()

conn.close()

return jsonify(asset\_list)

# Generate reports (simplified)

@app.route('/reports', methods=['GET'])

def generate\_reports():

# Example of simple report generation

reports = [

"Daily Report: 5 assets added.",

"Weekly Report: 20 assets updated.",

"Monthly Report: 100 assets in inventory.",

"Yearly Report: 1200 assets managed."

]

return jsonify(reports)

if \_\_name\_\_ == '\_\_main\_\_':

create\_asset\_table() # Ensure the table is created before the app runs

app.run(debug=True)

### ****5. PostgreSQL Setup****

Before running the Python app, make sure you have set up PostgreSQL with the following steps:

1. **Create the Database**:

CREATE DATABASE asset\_inventory;

2) **Create the Assets Table** (if needed, use the table creation command inside app.py):

CREATE TABLE assets (

id SERIAL PRIMARY KEY,

asset\_id VARCHAR(50) NOT NULL,

description TEXT NOT NULL,

comments TEXT,

asset\_type TEXT NOT NULL

);

**Project Directory Structure**

**asset\_inventory\_system/**

├── static/

│ ├── app.js

│ └── style.css

├── templates/

│ └── index.html

└── app.py

### ****nstructions to Run the Project****

1. **Install Python Dependencies**: Make sure you have installed the necessary Python libraries:

pip install Flask psycopg2

2) **Run the Flask Application**: Start the Flask application by running:

python app.py

**3) Access the Application**: Open your web browser and go to http://localhost:5000 to view the asset inventory management system.

### ****Conclusion****:

This basic implementation of an **Asset Inventory Management System** integrates a **PostgreSQL** database with **Flask** (Python), **HTML**, **CSS**, and **JavaScript**. The system allows you to add, delete, update assets, and generate inventory reports. You can further extend it with features like asset categorization, search, and advanced analytics.