Here’s a complete web application project that implements a simplified **Asset Inventory Management System** using **HTML**, **CSS**, **JavaScript**, and **Python (Flask)**. The system connects to a **PostgreSQL** database, performs CRUD operations for asset management, and generates reports on daily, weekly, monthly, and yearly basis.

**Project Structure**

1. **HTML** for the front-end structure.
2. **CSS** for styling.
3. **JavaScript** for client-side logic.
4. **Python (Flask)** for the backend.
5. **PostgreSQL** for database management.

**1. HTML - Asset Management Page (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>

<p>Manage your assets efficiently</p>

<!-- Asset Form -->

<form id="assetForm">

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

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

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

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

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

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

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

</form>

<h2>Asset Reports</h2>

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

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

<h2>Modify/Delete Asset</h2>

<button id="deleteAsset">Delete Asset</button>

<button id="modifyAsset">Modify Asset</button>

</div>

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

</body>

</html>

2. **CSS - Styling the Web Page (static/style.css)**

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

color: #333;

}

.container {

max-width: 700px;

margin: 50px auto;

padding: 20px;

background-color: #fff;

border-radius: 10px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

h1 {

text-align: center;

color: #4CAF50;

}

label {

margin-top: 10px;

display: block;

}

input {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

display: block;

width: 100%;

padding: 10px;

margin-top: 20px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #45a049;

}

#reportResults {

margin-top: 20px;

}

3. **JavaScript - 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,

};

fetch('/add-asset', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(assetData)

})

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

.then(data => {

alert('Asset added successfully!');

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

})

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

}

function generateReports() {

fetch('/get-reports')

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

.then(data => {

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

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

data.forEach(report => {

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

});

})

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

}

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

from flask import Flask, request, jsonify

from flask\_cors import CORS

import psycopg2

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

CORS(app)

# PostgreSQL connection

def get\_db\_connection():

conn = psycopg2.connect(

dbname='asset\_inventory',

user='yourUsername',

password='yourPassword',

host='localhost',

port='5432'

)

return conn

# Create table

def create\_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

);

''')

conn.commit()

cur.close()

conn.close()

# Add new asset

@app.route('/add-asset', methods=['POST'])

def add\_asset():

asset\_data = request.get\_json()

asset\_id = asset\_data['assetId']

description = asset\_data['description']

comments = asset\_data['comments']

conn = get\_db\_connection()

cur = conn.cursor()

cur.execute('''

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

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

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

conn.commit()

cur.close()

conn.close()

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

# Get reports

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

def get\_reports():

conn = get\_db\_connection()

cur = conn.cursor()

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

assets = cur.fetchall()

reports = []

for asset in assets:

reports.append(f'Asset ID: {asset[1]}, Description: {asset[2]}, Comments: {asset[3]}')

cur.close()

conn.close()

return jsonify(reports)

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

create\_table()

app.run(debug=True)

1. **PostgreSQL Database Setup**

SQL CODE:-

CREATE DATABASE asset\_inventory;

CREATE TABLE assets (

id SERIAL PRIMARY KEY,

asset\_id VARCHAR(50) NOT NULL,

description TEXT NOT NULL,

comments TEXT

);

**nstructions to Run the Project**

1. **Set up PostgreSQL**:
   * Install PostgreSQL and create a database named asset\_inventory.
   * Create the assets table using the provided SQL commands.
2. **Install Required Python Packages**:
   * Install Flask and psycopg2 using pip:

pip install Flask psycopg2 flask-cors

**3) Project Structure**:

* Create the following directory structure:

asset\_inventory\_management/

├── static/

│ ├── app.js

│ └── style.css

├── templates/

│ └── index.html

└── app.py

**4) Run the Flask App**:

* Navigate to the project directory and run:

python app.py

5) **Access the Web Page**:

* Open your web browser and go to http://127.0.0.1:5000 to access the application.

### Summary

This project is a basic implementation of an **Asset Inventory Management System**, using **HTML**, **CSS**, **JavaScript**, and **Python (Flask)**. It connects to a **PostgreSQL** database for managing assets and provides basic CRUD functionality along with report generation. You can modify and expand this project by adding advanced features like asset search, pagination, and filtering reports by date ranges.