9\_25\_24

Sdi\_biochar\_search1.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Soil Analysis Form</title>

</head>

<body>

<h1>Soil Analysis Form</h1>

<form method="post" action="/select\_crop">

<label for="moisture">Soil Moisture (%):</label>

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

<br><br>

<label for="organic\_matter">Organic Matter (%):</label>

<input type="text" id="organic\_matter" name="organic\_matter" required>

<br><br>

<label for="phosphorus">Phosphorus (ppm):</label>

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

<br><br>

<label for="potassium">Potassium (ppm):</label>

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

<br><br>

<label for="available\_N">Plant available N (either NH4+/NO3-) (ppm):</label>

<input type="text" id="available\_N" name="available\_N" required>

<br><br>

<label for="pH">pH:</label>

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

<br><br>

<label for="sand">Sand (%):</label>

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

<br><br>

<label for="silt">Silt (%):</label>

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

<br><br>

<label for="clay">Clay (%):</label>

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

<br><br>

<h2>Select Crop</h2>

<label for="crop">Choose a crop:</label>

<select name="crop" id="crop">

{% for crop in crops %}

<option value="{{ crop.Crop }}">{{ crop.Crop }}</option>

{% endfor %}

</select>

<br><br>

<input type="submit" value="Submit">

</form>

<!-- Display any messages from the server -->

{% if messages %}

<div>

{% for message in messages %}

<p>{{ message }}</p>

{% endfor %}

</div>

{% endif %}

{% if data %}

<h2>Submitted Data:</h2>

<ul>

{% for key, value in data.items() %}

<li>{{ key }}: {{ value }}</li>

{% endfor %}

</ul>

{% if messages %}

<h3>Messages:</h3>

<ul>

{% for key, value in messages.items() %}

<li>{{ value }}</li>

{% endfor %}

</ul>

{% endif %}

{% endif %}

<!-- Hidden fields to carry previously submitted data -->

<input type="hidden" name="moisture" value="{{ data['Soil Moisture (%)'] }}">

<input type="hidden" name="organic\_matter" value="{{ data['Organic Matter (%)'] }}">

<input type="hidden" name="phosphorus" value="{{ data['Phosphorus (ppm)'] }}">

<input type="hidden" name="potassium" value="{{ data['Potassium (ppm)'] }}">

<input type="hidden" name="available\_N" value="{{ data['Plant available Nitrogen (either NH4+/NO3-) (ppm)'] }}">

<input type="hidden" name="pH" value="{{ data['pH'] }}">

<input type="hidden" name="sand" value="{{ data['Sand (%)'] }}">

<input type="hidden" name="silt" value="{{ data['Silt (%)'] }}">

<input type="hidden" name="clay" value="{{ data['Clay (%)'] }}">

</form>

{% if crop\_data %}

<h2>Crop Requirements:</h2>

<ul>

<li>Required Phosphorus (lbs/acre): {{ crop\_data['Required Phosphorus (lbs/acre)'] }}</li>

<li>Required Potassium (lbs/acre): {{ crop\_data['Required Potassium (lbs/acre)'] }}</li>

<li>Required Nitrogen (lbs/acre): {{ crop\_data['Required Nitrogen (lbs/acre)'] }}</li>

{% if 'Required Lime (lbs/acre)' in crop\_data %}

<li>Required Lime (lbs/acre): {{ crop\_data['Required Lime (lbs/acre)'] }}</li>

{% endif %}

</ul>

{% endif %}

<!-- Display Biochar Recommendations -->

{% if biochar\_results %}

<h2>Top 5 Biochar Matches:</h2>

<h3>Phosphorus</h3>

<table border="1">

<tr>

<th>Sample Name</th>

<th>Extractable P (lbs/ton)</th>

</tr>

{% for sample in biochar\_results.phosphorus %}

<tr>

<td>{{ sample.Sample }}</td>

<td>{{ sample.ExtractablePlbs1ton }}</td>

</tr>

{% endfor %}

</table>

<h3>Potassium</h3>

<table border="1">

<tr>

<th>Sample Name</th>

<th>K (lbs/ton)</th>

</tr>

{% for sample in biochar\_results.potassium %}

<tr>

<td>{{ sample.Sample }}</td>

<td>{{ sample.Klb\_1ton }}</td>

</tr>

{% endfor %}

</table>

<h3>Nitrogen</h3>

<table border="1">

<tr>

<th>Sample Name</th>

<th>N (lbs/ton)</th>

</tr>

{% for sample in biochar\_results.nitrogen %}

<tr>

<td>{{ sample.Sample }}</td>

<td>{{ sample.Plant\_available\_Nlbs\_1ton }}</td>

</tr>

{% endfor %}

</table>

<h3>CaCO3eq</h3>

<table border="1">

<tr>

<th>Sample Name</th>

<th>CaCO3 (lbs/ton)</th>

</tr>

{% for sample in biochar\_results.lime %}

<tr>

<td>{{ sample.Sample }}</td>

<td>{{ sample.CaCO3lb\_1ton}}</td>

</tr>

{% endfor %}

</table>

{% endif %}

</body>

</html>

Sdi\_biochar\_search1.py

from flask import Flask, render\_template, request, redirect, url\_for

from flask\_sqlalchemy import SQLAlchemy

from sqlalchemy import asc, func

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

@app.route('/')

def index():

return redirect(url\_for('select\_crop'))

# Configure PostgreSQL database URI

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'postgresql://postgres:Postgres500!@localhost:5432/practiceAtlas'

db = SQLAlchemy(app)

# Define Crop model

class Crop(db.Model):

\_\_tablename\_\_ = 'Crop\_Fertilizer\_Guide'

Crop = db.Column(db.String(255), unique=True, nullable=False)

pH\_min = db.Column(db.Float)

pH\_max = db.Column(db.Float)

pH\_opt = db.Column(db.Float)

N\_low\_rate = db.Column(db.Float)

N\_upper\_rate = db.Column(db.Float)

P\_low\_rate = db.Column(db.Float)

P\_upper\_rate = db.Column(db.Float)

K\_low\_rate = db.Column(db.Float)

K\_upper\_rate = db.Column(db.Float)

Lime\_low\_rate = db.Column(db.Float)

Lime\_upper\_rate = db.Column(db.Float)

Reference = db.Column(db.String(255))

ID = db.Column(db.Integer, primary\_key=True)

# Define Biochar models

class ExtractableP(db.Model):

\_\_tablename\_\_ = 'Extractable P'

ID = db.Column(db.Integer, primary\_key=True)

Sample = db.Column(db.String(255), unique=True, nullable=False)

Extractable\_P\_mean = db.Column(db.Float)

Extractable\_P\_std = db.Column(db.Float)

ExtractablePlbs1ton = db.Column(db.Float)

class ExtractableNutrients(db.Model):

\_\_tablename\_\_ = 'Extractable Nutrients'

ID = db.Column(db.Integer, primary\_key=True)

Sample = db.Column(db.String(255), unique=True, nullable=False)

Na\_ave = db.Column(db.Float)

Na\_std = db.Column(db.Float)

Nalb\_1ton = db.Column(db.Float)

K\_ave = db.Column(db.Float)

K\_std = db.Column(db.Float)

Klb\_1ton = db.Column(db.Float)

Mg\_ave = db.Column(db.Float)

Mg\_std = db.Column(db.Float)

Mglb\_1ton = db.Column(db.Float)

Ca\_ave = db.Column(db.Float)

Ca\_std = db.Column(db.Float)

Calb\_1ton = db.Column(db.Float)

S\_ave = db.Column(db.Float)

S\_std = db.Column(db.Float)

Slb\_1ton = db.Column(db.Float)

class PlantAvailableN(db.Model):

\_\_tablename\_\_ = 'Plant Available N'

ID = db.Column(db.Integer, primary\_key=True)

Sample = db.Column(db.String(255), unique=True, nullable=False)

reported\_NH4\_mean = db.Column(db.Float)

reported\_NH4\_std = db.Column(db.Float)

reported\_NO3\_mean = db.Column(db.Float)

reported\_NO3\_std = db.Column(db.Float)

Plant\_available\_N = db.Column(db.Float)

Plant\_available\_Nlbs\_1ton = db.Column(db.Float)

class CaCO3Eq(db.Model):

\_\_tablename\_\_ = 'CaCO3-eq'

ID = db.Column(db.Integer, primary\_key=True)

Sample = db.Column(db.String(255), unique=True, nullable=False)

CaCO3eq\_mean = db.Column(db.Float)

CaCO3eq\_STD = db.Column(db.Float)

CaCO3lb\_1ton = db.Column(db.Float)

@app.route('/select\_crop', methods=['GET', 'POST'])

def select\_crop():

crops = Crop.query.all()

messages = {}

data = None

soil\_data = {}

biochar\_results = None

if request.method == 'POST':

selected\_crop = request.form['crop']

soil\_data = {

"Soil Moisture (%)": request.form.get("moisture"),

"Organic Matter (%)": request.form.get("organic\_matter"),

"Phosphorus (ppm)": request.form.get("phosphorus"),

"Potassium (ppm)": request.form.get("potassium"),

"Plant available Nitrogen (either NH4+/NO3-) (ppm)": request.form.get("available\_N"),

"pH": request.form.get("pH"),

"Sand (%)": request.form.get("sand"),

"Silt (%)": request.form.get("silt"),

"Clay (%)": request.form.get("clay")

}

crop = Crop.query.filter\_by(Crop=selected\_crop).first()

if crop:

# Calculate nutrient needs

phosphorus\_needed = max(0, crop.P\_low\_rate - (float(soil\_data["Phosphorus (ppm)"]) \* 2.2913))

potassium\_needed = max(0, crop.K\_low\_rate - (float(soil\_data["Potassium (ppm)"]) \* 1.2046))

nitrogen\_needed = max(0, crop.N\_low\_rate - (float(soil\_data["Plant available Nitrogen (either NH4+/NO3-) (ppm)"]) \* 2))

lime\_needed = 0

if float(soil\_data["pH"]) < crop.pH\_min:

lime\_needed = crop.Lime\_low\_rate

# Fetch top 5 biochar samples based on closest values to nutrient needs

top\_phosphorus = ExtractableP.query.order\_by(func.abs(ExtractableP.ExtractablePlbs1ton - phosphorus\_needed)).limit(5).all()

top\_potassium = ExtractableNutrients.query.order\_by(func.abs(ExtractableNutrients.Klb\_1ton - potassium\_needed)).limit(5).all()

top\_nitrogen = PlantAvailableN.query.order\_by(func.abs(PlantAvailableN.Plant\_available\_Nlbs\_1ton - nitrogen\_needed)).limit(5).all()

top\_lime = CaCO3Eq.query.order\_by(func.abs(CaCO3Eq.CaCO3lb\_1ton - lime\_needed)).limit(5).all()

print("Top Phosphorus Matches:", top\_phosphorus)

print("Top Potassium Matches:", top\_potassium)

print("Top Nitrogen Matches:", top\_nitrogen)

print("Top Lime Matches:", top\_lime)

# Compile the data for rendering

biochar\_results = {

"phosphorus": top\_phosphorus,

"potassium": top\_potassium,

"nitrogen": top\_nitrogen,

"lime": top\_lime

}

# Display the required nutrient amounts to the user

data = {

"Crop": selected\_crop,

"Required Phosphorus (lbs/acre)": phosphorus\_needed,

"Required Potassium (lbs/acre)": potassium\_needed,

"Required Nitrogen (lbs/acre)": nitrogen\_needed,

}

if lime\_needed > 0:

data["Required Lime (lbs/acre)"] = lime\_needed

else:

messages['error'] = "Crop not found."

return render\_template('sdi\_biochar\_search1.html', data=soil\_data, messages=messages, crops=crops, crop\_data=data, biochar\_results=biochar\_results)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)