In [2]:
```
pip install geopandas
```

```
Collecting geopandas
  Obtaining dependency information for geopandas from https://files.pythonho
sted.org/packages/c4/64/7d344cfcef5efddf9cf32f59af7f855828e9d74b5f862eddf5bf
d9f25323/geopandas-1.0.1-py3-none-any.whl.metadata
  Downloading geopandas-1.0.1-py3-none-any.whl.metadata (2.2 kB)
Requirement already satisfied: numpy>=1.22 in ./anaconda3/lib/python3.11/sit
e-packages (from geopandas) (1.24.3)
Collecting pyogrio>=0.7.2 (from geopandas)
  Obtaining dependency information for pyogrio>=0.7.2 from https://files.pyt
honhosted.org/packages/8d/2c/c761e6adeb81bd4029a137b3240e7214a8c9aaf22588335
6196afd6ef9d8/pyogrio-0.10.0-cp311-cp311-macosx_12_0_arm64.whl.metadata
  Downloading pyogrio-0.10.0-cp311-cp311-macosx_12_0_arm64.whl.metadata (5.5
kB)
Requirement already satisfied: packaging in ./anaconda3/lib/python3.11/site-
packages (from geopandas) (23.0)
Requirement already satisfied: pandas>=1.4.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (1.5.3)
Collecting pyproj>=3.3.0 (from geopandas)
  Obtaining dependency information for pyproj>=3.3.0 from https://files.pyth
onhosted.org/packages/2d/4d/610fe2a17de71b4fe210af69ce25f2d65379ba0a48299129
894d0d0988ee/pyproj-3.7.0-cp311-cp311-macosx_14_0_arm64.whl.metadata
  Downloading pyproj-3.7.0-cp311-cp311-macosx_14_0_arm64.whl.metadata (31 k
B)
Collecting shapely>=2.0.0 (from geopandas)
  Obtaining dependency information for shapely>=2.0.0 from https://files.pyt
honhosted.org/packages/37/63/e182e43081fffa0a2d970c480f2ef91647a6ab94098f617
48c23c2a485f2/shapely-2.0.6-cp311-cp311-macosx_11_0_arm64.whl.metadata
  Downloading shapely-2.0.6-cp311-cp311-macosx_11_0_arm64.whl.metadata (7.0
kB)
Requirement already satisfied: python-dateutil>=2.8.1 in ./anaconda3/lib/pyt
hon3.11/site-packages (from pandas>=1.4.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/si
te-packages (from pandas>=1.4.0->geopandas) (2022.7)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from pyogrio>=0.7.2->geopandas) (2023.7.22)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-p
ackages (from python-dateutil>=2.8.1->pandas>=1.4.0->geopandas) (1.16.0)
Downloading geopandas-1.0.1-py3-none-any.whl (323 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.6/323.6 kB 685.1 kB/s eta 0:
00:00a 0:00:01
Downloading pyogrio-0.10.0-cp311-cp311-macosx_12_0_arm64.whl (15.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.1/15.1 MB 2.1 MB/s eta 0:00:
0000:0100:01m
Downloading pyproj-3.7.0-cp311-cp311-macosx_14_0_arm64.whl (4.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.6/4.6 MB 7.9 MB/s eta 0:00:00
00:0100:01
Downloading shapely-2.0.6-cp311-cp311-macosx_11_0_arm64.whl (1.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 10.2 MB/s eta 0:00:0
0a 0:00:01
Installing collected packages: shapely, pyproj, pyogrio, geopandas
Successfully installed geopandas-1.0.1 pyogrio-0.10.0 pyproj-3.7.0 shapely-
2.0.6
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install fiona shapely pyproj rtree
```

```
Collecting fiona
  Obtaining dependency information for fiona from https://files.pythonhoste
d.org/packages/65/0c/e8070b15c8303f60bd4444a120842597ccd6ed550548948e2e36cff
baa93/fiona-1.10.1-cp311-cp311-macosx_11_0_arm64.whl.metadata
  Downloading fiona-1.10.1-cp311-cp311-macosx_11_0_arm64.whl.metadata (56 k
B)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.6/56.6 kB 1.5 MB/s eta 0:0
0:00a 0:00:01
Requirement already satisfied: shapely in ./anaconda3/lib/python3.11/site-pa
ckages (2.0.6)
Requirement already satisfied: pyproj in ./anaconda3/lib/python3.11/site-pac
kages (3.7.0)
Requirement already satisfied: rtree in ./anaconda3/lib/python3.11/site-pack
ages (1.0.1)
Requirement already satisfied: attrs>=19.2.0 in ./anaconda3/lib/python3.11/s
ite-packages (from fiona) (23.2.0)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from fiona) (2023.7.22)
Requirement already satisfied: click~=8.0 in ./anaconda3/lib/python3.11/site
-packages (from fiona) (8.0.4)
Collecting click-plugins>=1.0 (from fiona)
  Obtaining dependency information for click-plugins>=1.0 from https://file
s.pythonhosted.org/packages/e9/da/824b92d9942f4e472702488857914bdd50f73021ef
ea15b4cad9aca8ecef/click_plugins-1.1.1-py2.py3-none-any.whl.metadata
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl.metadata (6.4 kB)
Collecting cligj>=0.5 (from fiona)
  Obtaining dependency information for cligj>=0.5 from https://files.pythonh
osted.org/packages/73/86/43fa9f15c5b9fb6e82620428827cd3c284aa933431405d1bcf5
231ae3d3e/cligj-0.7.2-py3-none-any.whl.metadata
  Downloading cligj-0.7.2-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: numpy<3,>=1.14 in ./anaconda3/lib/python3.11/
site-packages (from shapely) (1.24.3)
Downloading fiona-1.10.1-cp311-cp311-macosx_11_0_arm64.whl (14.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.8/14.8 MB 4.1 MB/s eta 0:00:
0000:0100:01
Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Installing collected packages: cligj, click-plugins, fiona
Successfully installed click-plugins-1.1.1 cligj-0.7.2 fiona-1.10.1
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
pip install geopandas shapely
```

```
Requirement already satisfied: geopandas in ./anaconda3/lib/python3.11/site-
packages (1.0.1)
Requirement already satisfied: shapely in ./anaconda3/lib/python3.11/site-pa
ckages (2.0.6)
Requirement already satisfied: numpy>=1.22 in ./anaconda3/lib/python3.11/sit
e-packages (from geopandas) (1.24.3)
Requirement already satisfied: pyogrio>=0.7.2 in ./anaconda3/lib/python3.11/
site-packages (from geopandas) (0.10.0)
Requirement already satisfied: packaging in ./anaconda3/lib/python3.11/site-
packages (from geopandas) (23.0)
Requirement already satisfied: pandas>=1.4.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (1.5.3)
Requirement already satisfied: pyproj>=3.3.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (3.7.0)
Requirement already satisfied: python-dateutil>=2.8.1 in ./anaconda3/lib/pyt
hon3.11/site-packages (from pandas>=1.4.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/si
te-packages (from pandas>=1.4.0->geopandas) (2022.7)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from pyogrio>=0.7.2->geopandas) (2023.7.22)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-p
ackages (from python-dateutil>=2.8.1->pandas>=1.4.0->geopandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [14]:
```
pip install rasterio numpy geopandas
```

```
Collecting rasterio
  Obtaining dependency information for rasterio from https://files.pythonhos
ted.org/packages/b3/59/ca86697161206233eea6353237b0c0f02f6f70434144db162f964
a7e1b19/rasterio-1.4.3-cp311-cp311-macosx_14_0_arm64.whl.metadata
  Downloading rasterio-1.4.3-cp311-cp311-macosx_14_0_arm64.whl.metadata (9.1
kB)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-pack
ages (1.24.3)
Requirement already satisfied: geopandas in ./anaconda3/lib/python3.11/site-
packages (1.0.1)
Collecting affine (from rasterio)
  Obtaining dependency information for affine from https://files.pythonhoste
d.org/packages/0b/f7/85273299ab57117850cc0a936c64151171fac4da49bc6fba0dad984
a7c5f/affine-2.4.0-py3-none-any.whl.metadata
  Downloading affine-2.4.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: attrs in ./anaconda3/lib/python3.11/site-pack
ages (from rasterio) (23.2.0)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from rasterio) (2023.7.22)
Requirement already satisfied: click>=4.0 in ./anaconda3/lib/python3.11/site
-packages (from rasterio) (8.0.4)
Requirement already satisfied: cligj>=0.5 in ./anaconda3/lib/python3.11/site
-packages (from rasterio) (0.7.2)
Requirement already satisfied: click-plugins in ./anaconda3/lib/python3.11/s
ite-packages (from rasterio) (1.1.1)
Requirement already satisfied: pyparsing in ./anaconda3/lib/python3.11/site-
packages (from rasterio) (3.0.9)
Requirement already satisfied: pyogrio>=0.7.2 in ./anaconda3/lib/python3.11/
site-packages (from geopandas) (0.10.0)
Requirement already satisfied: packaging in ./anaconda3/lib/python3.11/site-
packages (from geopandas) (23.0)
Requirement already satisfied: pandas>=1.4.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (1.5.3)
Requirement already satisfied: pyproj>=3.3.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (3.7.0)
Requirement already satisfied: shapely>=2.0.0 in ./anaconda3/lib/python3.11/
site-packages (from geopandas) (2.0.6)
Requirement already satisfied: python-dateutil>=2.8.1 in ./anaconda3/lib/pyt
hon3.11/site-packages (from pandas>=1.4.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/si
te-packages (from pandas>=1.4.0->geopandas) (2022.7)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-p
ackages (from python-dateutil>=2.8.1->pandas>=1.4.0->geopandas) (1.16.0)
Downloading rasterio-1.4.3-cp311-cp311-macosx_14_0_arm64.whl (18.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.8/18.8 MB 317.6 kB/s eta 0:0
0:0000:0100:02
Downloading affine-2.4.0-py3-none-any.whl (15 kB)
Installing collected packages: affine, rasterio
Successfully installed affine-2.4.0 rasterio-1.4.3
Note: you may need to restart the kernel to use updated packages.
```

In [18]:
```
pip install numpy rasterio
```

```
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-pack
ages (1.24.3)
Requirement already satisfied: rasterio in ./anaconda3/lib/python3.11/site-p
ackages (1.4.3)
Requirement already satisfied: affine in ./anaconda3/lib/python3.11/site-pac
kages (from rasterio) (2.4.0)
Requirement already satisfied: attrs in ./anaconda3/lib/python3.11/site-pack
ages (from rasterio) (23.2.0)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from rasterio) (2023.7.22)
Requirement already satisfied: click>=4.0 in ./anaconda3/lib/python3.11/site
-packages (from rasterio) (8.0.4)
Requirement already satisfied: cligj>=0.5 in ./anaconda3/lib/python3.11/site
-packages (from rasterio) (0.7.2)
Requirement already satisfied: click-plugins in ./anaconda3/lib/python3.11/s
ite-packages (from rasterio) (1.1.1)
Requirement already satisfied: pyparsing in ./anaconda3/lib/python3.11/site-
packages (from rasterio) (3.0.9)
Note: you may need to restart the kernel to use updated packages.
```

In [43]:
```
pip install folium
```

```
Collecting folium
  Obtaining dependency information for folium from https://files.pythonhoste
d.org/packages/fc/ab/d1f47c48a14e17cd487c8b467b573291fae75477b067241407e7889
a3692/folium-0.19.4-py2.py3-none-any.whl.metadata
  Downloading folium-0.19.4-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting branca>=0.6.0 (from folium)
  Obtaining dependency information for branca>=0.6.0 from https://files.pyth
onhosted.org/packages/f8/9d/91cddd38bd00170aad1a4b198c47b4ed716be45c234e09b8
35af41f4e717/branca-0.8.1-py3-none-any.whl.metadata
  Downloading branca-0.8.1-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: jinja2>=2.9 in ./anaconda3/lib/python3.11/sit
e-packages (from folium) (3.1.2)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-pack
ages (from folium) (1.24.3)
Requirement already satisfied: requests in ./anaconda3/lib/python3.11/site-p
ackages (from folium) (2.31.0)
Requirement already satisfied: xyzservices in ./anaconda3/lib/python3.11/sit
e-packages (from folium) (2022.9.0)
Requirement already satisfied: MarkupSafe>=2.0 in ./anaconda3/lib/python3.1
1/site-packages (from jinja2>=2.9->folium) (2.1.1)
Requirement already satisfied: charset-normalizer<4,>=2 in ./anaconda3/lib/p
ython3.11/site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./anaconda3/lib/python3.11/si
te-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./anaconda3/lib/python
3.11/site-packages (from requests->folium) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in ./anaconda3/lib/python
3.11/site-packages (from requests->folium) (2023.7.22)
Downloading folium-0.19.4-py2.py3-none-any.whl (110 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 924.8 kB/s eta 0:
00:00a 0:00:01
Downloading branca-0.8.1-py3-none-any.whl (26 kB)
Installing collected packages: branca, folium
Successfully installed branca-0.8.1 folium-0.19.4
Note: you may need to restart the kernel to use updated packages.
```

In [50]:
```
pip install numpy pandas
```

```
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-pack
ages (1.24.3)
Requirement already satisfied: pandas in ./anaconda3/lib/python3.11/site-pac
kages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in ./anaconda3/lib/pyt
hon3.11/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/si
te-packages (from pandas) (2022.7)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-p
ackages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [62]:
```
pip install numpy rasterio matplotlib
```

```
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-pack
ages (1.24.3)
Requirement already satisfied: rasterio in ./anaconda3/lib/python3.11/site-p
ackages (1.4.3)
Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.11/site
-packages (3.7.1)
Requirement already satisfied: affine in ./anaconda3/lib/python3.11/site-pac
kages (from rasterio) (2.4.0)
Requirement already satisfied: attrs in ./anaconda3/lib/python3.11/site-pack
ages (from rasterio) (23.2.0)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from rasterio) (2023.7.22)
Requirement already satisfied: click>=4.0 in ./anaconda3/lib/python3.11/site
-packages (from rasterio) (8.0.4)
Requirement already satisfied: cligj>=0.5 in ./anaconda3/lib/python3.11/site
-packages (from rasterio) (0.7.2)
Requirement already satisfied: click-plugins in ./anaconda3/lib/python3.11/s
ite-packages (from rasterio) (1.1.1)
Requirement already satisfied: pyparsing in ./anaconda3/lib/python3.11/site-
packages (from rasterio) (3.0.9)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/python3.1
1/site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.11/si
te-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/python3.
11/site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.
11/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in ./anaconda3/lib/python3.1
1/site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in ./anaconda3/lib/python3.11/s
ite-packages (from matplotlib) (9.4.0)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/lib/pytho
n3.11/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-p
ackages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [68]:
```
!pip install jenkspy
```

```
Requirement already satisfied: jenkspy in ./anaconda3/lib/python3.11/site-pa
ckages (0.4.1)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-pack
ages (from jenkspy) (1.24.3)
```

In [77]:
```
pip install geopandas shapely matplotlib
```

```
Requirement already satisfied: geopandas in ./anaconda3/lib/python3.11/site-
packages (1.0.1)
Requirement already satisfied: shapely in ./anaconda3/lib/python3.11/site-pa
ckages (2.0.6)
Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.11/site
-packages (3.7.1)
Requirement already satisfied: numpy>=1.22 in ./anaconda3/lib/python3.11/sit
e-packages (from geopandas) (1.24.3)
Requirement already satisfied: pyogrio>=0.7.2 in ./anaconda3/lib/python3.11/
site-packages (from geopandas) (0.10.0)
Requirement already satisfied: packaging in ./anaconda3/lib/python3.11/site-
packages (from geopandas) (23.0)
Requirement already satisfied: pandas>=1.4.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (1.5.3)
Requirement already satisfied: pyproj>=3.3.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (3.7.0)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/python3.1
1/site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.11/si
te-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/python3.
11/site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.
11/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pillow>=6.2.0 in ./anaconda3/lib/python3.11/s
ite-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in ./anaconda3/lib/python3.1
1/site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/lib/pytho
n3.11/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/si
te-packages (from pandas>=1.4.0->geopandas) (2022.7)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from pyogrio>=0.7.2->geopandas) (2023.7.22)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-p
ackages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [84]:
```
pip install streamlit geopandas matplotlib rasterio
```

```
Collecting streamlit
  Obtaining dependency information for streamlit from https://files.pythonho
sted.org/packages/c2/87/b2e162869500062a94dde7589c167367b5538dab6eacce2e7c0f
00d5c9c5/streamlit-1.41.1-py2.py3-none-any.whl.metadata
  Downloading streamlit-1.41.1-py2.py3-none-any.whl.metadata (8.5 kB)
Requirement already satisfied: geopandas in ./anaconda3/lib/python3.11/site-
packages (1.0.1)
Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.11/site
-packages (3.7.1)
Requirement already satisfied: rasterio in ./anaconda3/lib/python3.11/site-p
```

```
ackages (1.4.3)
Collecting altair<6,>=4.0 (from streamlit)
  Obtaining dependency information for altair<6,>=4.0 from https://files.pyt
honhosted.org/packages/aa/f3/0b6ced594e51cc95d8c1fc1640d3623770d01e4969d29c0
bd09945fafefa/altair-5.5.0-py3-none-any.whl.metadata
  Downloading altair-5.5.0-py3-none-any.whl.metadata (11 kB)
Collecting blinker<2,>=1.0.0 (from streamlit)
  Obtaining dependency information for blinker<2,>=1.0.0 from https://files.
pythonhosted.org/packages/10/cb/f2ad4230dc2eb1a74edf38f1a38b9b52277f75bef262
d8908e60d957e13c/blinker-1.9.0-py3-none-any.whl.metadata
  Downloading blinker-1.9.0-py3-none-any.whl.metadata (1.6 kB)
Collecting cachetools<6,>=4.0 (from streamlit)
  Obtaining dependency information for cachetools<6,>=4.0 from https://file
s.pythonhosted.org/packages/a4/07/14f8ad37f2d12a5ce41206c21820d8cb6561b728e5
1fad4530dff0552a67/cachetools-5.5.0-py3-none-any.whl.metadata
  Downloading cachetools-5.5.0-py3-none-any.whl.metadata (5.3 kB)
Requirement already satisfied: click<9,>=7.0 in ./anaconda3/lib/python3.11/s
ite-packages (from streamlit) (8.0.4)
Requirement already satisfied: numpy<3,>=1.23 in ./anaconda3/lib/python3.11/
site-packages (from streamlit) (1.24.3)
Requirement already satisfied: packaging<25,>=20 in ./anaconda3/lib/python3.
11/site-packages (from streamlit) (23.0)
Requirement already satisfied: pandas<3,>=1.4.0 in ./anaconda3/lib/python3.1
1/site-packages (from streamlit) (1.5.3)
Requirement already satisfied: pillow<12,>=7.1.0 in ./anaconda3/lib/python3.
11/site-packages (from streamlit) (9.4.0)
Requirement already satisfied: protobuf<6,>=3.20 in ./anaconda3/lib/python3.
11/site-packages (from streamlit) (4.25.4)
Requirement already satisfied: pyarrow>=7.0 in ./anaconda3/lib/python3.11/si
te-packages (from streamlit) (11.0.0)
Requirement already satisfied: requests<3,>=2.27 in ./anaconda3/lib/python3.
11/site-packages (from streamlit) (2.31.0)
Requirement already satisfied: rich<14,>=10.14.0 in ./anaconda3/lib/python3.
11/site-packages (from streamlit) (13.8.0)
Requirement already satisfied: tenacity<10,>=8.1.0 in ./anaconda3/lib/python
3.11/site-packages (from streamlit) (8.2.2)
Requirement already satisfied: toml<2,>=0.10.1 in ./anaconda3/lib/python3.1
1/site-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in ./anaconda3/li
b/python3.11/site-packages (from streamlit) (4.11.0)
Collecting gitpython!=3.1.19,<4,>=3.0.7 (from streamlit)
  Obtaining dependency information for gitpython!=3.1.19,<4,>=3.0.7 from htt
ps://files.pythonhosted.org/packages/1d/9a/4114a9057db2f1462d5c8f8390ab73839
25fe1ac012eaa42402ad65c2963/GitPython-3.1.44-py3-none-any.whl.metadata
  Downloading GitPython-3.1.44-py3-none-any.whl.metadata (13 kB)
Collecting pydeck<1,>=0.8.0b4 (from streamlit)
  Obtaining dependency information for pydeck<1,>=0.8.0b4 from https://file
s.pythonhosted.org/packages/ab/4c/b888e6cf58bd9db9c93f40d1c6be8283ff49d88919
231afe93a6bcf61626/pydeck-0.9.1-py2.py3-none-any.whl.metadata
  Downloading pydeck-0.9.1-py2.py3-none-any.whl.metadata (4.1 kB)
Requirement already satisfied: tornado<7,>=6.0.3 in ./anaconda3/lib/python3.
11/site-packages (from streamlit) (6.3.2)
Requirement already satisfied: pyogrio>=0.7.2 in ./anaconda3/lib/python3.11/
site-packages (from geopandas) (0.10.0)
Requirement already satisfied: pyproj>=3.3.0 in ./anaconda3/lib/python3.11/s
ite-packages (from geopandas) (3.7.0)
Requirement already satisfied: shapely>=2.0.0 in ./anaconda3/lib/python3.11/
```

```
site-packages (from geopandas) (2.0.6)
Requirement already satisfied: contourpy>=1.0.1 in ./anaconda3/lib/python3.1
1/site-packages (from matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.11/si
te-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in ./anaconda3/lib/python3.
11/site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.
11/site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in ./anaconda3/lib/python3.1
1/site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in ./anaconda3/lib/pytho
n3.11/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: affine in ./anaconda3/lib/python3.11/site-pac
kages (from rasterio) (2.4.0)
Requirement already satisfied: attrs in ./anaconda3/lib/python3.11/site-pack
ages (from rasterio) (23.2.0)
Requirement already satisfied: certifi in ./anaconda3/lib/python3.11/site-pa
ckages (from rasterio) (2023.7.22)
Requirement already satisfied: cligj>=0.5 in ./anaconda3/lib/python3.11/site
-packages (from rasterio) (0.7.2)
Requirement already satisfied: click-plugins in ./anaconda3/lib/python3.11/s
ite-packages (from rasterio) (1.1.1)
Requirement already satisfied: jinja2 in ./anaconda3/lib/python3.11/site-pac
kages (from altair<6,>=4.0->streamlit) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in ./anaconda3/lib/python3.1
1/site-packages (from altair<6,>=4.0->streamlit) (4.17.3)
Collecting narwhals>=1.14.2 (from altair<6,>=4.0->streamlit)
  Obtaining dependency information for narwhals>=1.14.2 from https://files.p
ythonhosted.org/packages/17/d9/00a937201a3ca6e8b2ca29226935be53dd7bc9bee27cb
8ba5b27efe11ccf/narwhals-1.22.0-py3-none-any.whl.metadata
  Downloading narwhals-1.22.0-py3-none-any.whl.metadata (10.0 kB)
Collecting gitdb<5,>=4.0.1 (from gitpython!=3.1.19,<4,>=3.0.7->streamlit)
  Obtaining dependency information for gitdb<5,>=4.0.1 from https://files.py
thonhosted.org/packages/a0/61/5c78b91c3143ed5c14207f463aecfc8f9dbb5092fb2869
baf37c273b2705/gitdb-4.0.12-py3-none-any.whl.metadata
  Downloading gitdb-4.0.12-py3-none-any.whl.metadata (1.2 kB)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/si
te-packages (from pandas<3,>=1.4.0->streamlit) (2022.7)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-p
ackages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in ./anaconda3/lib/p
ython3.11/site-packages (from requests<3,>=2.27->streamlit) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in ./anaconda3/lib/python3.11/si
te-packages (from requests<3,>=2.27->streamlit) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in ./anaconda3/lib/python
3.11/site-packages (from requests<3,>=2.27->streamlit) (1.26.16)
Requirement already satisfied: markdown-it-py>=2.2.0 in ./anaconda3/lib/pyth
on3.11/site-packages (from rich<14,>=10.14.0->streamlit) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in ./anaconda3/lib/py
thon3.11/site-packages (from rich<14,>=10.14.0->streamlit) (2.15.1)
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.
0.7->streamlit)
  Obtaining dependency information for smmap<6,>=3.0.1 from https://files.py
thonhosted.org/packages/04/be/d09147ad1ec7934636ad912901c5fd7667e1c858e19d35
5237db0d0cd5e4/smmap-5.0.2-py3-none-any.whl.metadata
  Downloading smmap-5.0.2-py3-none-any.whl.metadata (4.3 kB)
```

```
Requirement already satisfied: MarkupSafe>=2.0 in ./anaconda3/lib/python3.1
1/site-packages (from jinja2->altair<6,>=4.0->streamlit) (2.1.1)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0
in ./anaconda3/lib/python3.11/site-packages (from jsonschema>=3.0->altair<6,
>=4.0->streamlit) (0.18.0)
Requirement already satisfied: mdurl~=0.1 in ./anaconda3/lib/python3.11/site
-packages (from markdown-it-py>=2.2.0->rich<14,>=10.14.0->streamlit) (0.1.0)
Downloading streamlit-1.41.1-py2.py3-none-any.whl (9.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.1/9.1 MB 1.7 MB/s eta 0:00:00
00:0100:010m
Downloading altair-5.5.0-py3-none-any.whl (731 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 731.2/731.2 kB 9.8 MB/s eta 0:0
0:00a 0:00:01
Downloading blinker-1.9.0-py3-none-any.whl (8.5 kB)
Downloading cachetools-5.5.0-py3-none-any.whl (9.5 kB)
Downloading GitPython-3.1.44-py3-none-any.whl (207 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.6/207.6 kB 6.3 MB/s eta 0:0
0:00
Downloading pydeck-0.9.1-py2.py3-none-any.whl (6.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 12.8 MB/s eta 0:00:0
000:0100:01
Downloading gitdb-4.0.12-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.8/62.8 kB 7.9 MB/s eta 0:00:
00
Downloading narwhals-1.22.0-py3-none-any.whl (297 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 297.5/297.5 kB 15.2 MB/s eta 0:
00:00
Downloading smmap-5.0.2-py3-none-any.whl (24 kB)
Installing collected packages: smmap, narwhals, cachetools, blinker, pydeck,
gitdb, altair, gitpython, streamlit
Successfully installed altair-5.5.0 blinker-1.9.0 cachetools-5.5.0 gitdb-4.
0.12 gitpython-3.1.44 narwhals-1.22.0 pydeck-0.9.1 smmap-5.0.2 streamlit-1.4
1.1
Note: you may need to restart the kernel to use updated packages.
```

In [85]:

```
import geopandas as gpd
import pandas as pd
from shapely.geometry import Point, LineString, Polygon
import numpy as np
import rasterio
from rasterio.transform import from_origin
import os
from rasterio.warp import calculate_default_transform, reproject, Resampling
import matplotlib.pyplot as plt
import folium
from rasterio.plot import reshape_as_image
import seaborn as sns
from matplotlib import patches
import jenkspy
import streamlit as st
```

In [24]:

```python
# Grid properties
width, height = 100, 100
cell_size = 0.01
x_min, y_max = 29.0, 39.8  # Top-left corner of the grid

# Synthetic raster data
lkp_layer = np.random.rand(height, width)  # LKP likelihood layer

# Transform and save the raster
transform = from_origin(x_min, y_max, cell_size, cell_size)

with rasterio.open(
    "synthetic_lkp_raster.tif",
    "w",
    driver="GTiff",
    height=height,
    width=width,
    count=1,
    dtype=lkp_layer.dtype,
    crs="EPSG:4326",
    transform=transform,
) as dst:
    dst.write(lkp_layer, 1)

print("Synthetic raster saved as 'synthetic_lkp_raster.tif'")
```

```
Synthetic raster saved as 'synthetic_lkp_raster.tif'
```

In [13]:
```
# Ensure the directory exists
output_dir = "./output_data"
os.makedirs(output_dir, exist_ok=True)

# Step 1: Create settlement points (villages)
settlements_data = {
    "name": ["Village A", "Village B", "Village C"],
    "geometry": [Point(29.3, 39.4), Point(29.4, 39.5), Point(29.5, 39.3)],
}
settlements = gpd.GeoDataFrame(settlements_data, crs="EPSG:4326")  # WGS84 C

# Step 2: Reproject to a projected CRS for buffer calculation (e.g., UTM Zon
settlements = settlements.to_crs(epsg=32635)

# Step 3: Add buffers around settlements (e.g., 5 km buffer)
settlements["buffer"] = settlements.geometry.buffer(5000)  # Buffer in meter

# Create a separate GeoDataFrame for buffers
settlements_buffers = settlements.copy()
settlements_buffers = settlements_buffers.set_geometry("buffer").drop(column

# Step 4: Create road lines in the same CRS
roads_data = {
    "name": ["Road 1", "Road 2"],
    "geometry": [
        LineString([(629000, 4350000), (635000, 4390000)]),
        LineString([(631000, 4350000), (631000, 4390000)]),
    ],
}
roads = gpd.GeoDataFrame(roads_data, crs="EPSG:32635")

# Step 5: Create a region of interest (polygon)
region_polygon = Polygon([(628000, 4340000), (637000, 4340000), (637000, 440
region = gpd.GeoDataFrame({"name": ["Region of Interest"], "geometry": [regi

# Step 6: Save the data to files
settlements.to_crs(epsg=4326).drop(columns=["buffer"]).to_file(f"{output_dir
settlements_buffers.to_crs(epsg=4326).to_file(f"{output_dir}/settlements_wit
roads.to_crs(epsg=4326).to_file(f"{output_dir}/roads.shp")  # Save roads
region.to_crs(epsg=4326).to_file(f"{output_dir}/region.shp")  # Save region

f"Shapefiles saved successfully in {output_dir}."
```

Out[13]:
```
'Shapefiles saved successfully in ./output_data.'
```

In [17]:
```python
# Step 1: Define grid properties
width, height = 100, 100  # Grid size
cell_size = 0.01  # Cell size in degrees
x_min, y_max = 29.0, 39.8  # Top-left corner of the grid

# Step 2: Create raster layers for criteria
# Example: LKP influence (high probability near center)
x_center, y_center = 29.4, 39.5
x = np.linspace(x_min, x_min + width * cell_size, width)
y = np.linspace(y_max, y_max - height * cell_size, height)
xv, yv = np.meshgrid(x, y)

# Compute distance from LKP
lkp_layer = np.exp(-np.sqrt((xv - x_center)**2 + (yv - y_center)**2) / 0.05)

# Example: Settlement influence (higher probability in certain areas)
settlement_layer = np.zeros((height, width))
settlement_coords = [(29.3, 39.4), (29.5, 39.3)]
for sx, sy in settlement_coords:
    settlement_layer += np.exp(-np.sqrt((xv - sx)**2 + (yv - sy)**2) / 0.03)

# Example: Terrain influence (random example values)
terrain_layer = np.random.rand(height, width)

# Step 3: Weighted overlay
weights = {
    "lkp": 0.5,
    "settlement": 0.3,
    "terrain": 0.2,
}
combined_layer = (weights["lkp"] * lkp_layer +
                  weights["settlement"] * settlement_layer +
                  weights["terrain"] * terrain_layer)

# Step 4: Save the probability map as a GeoTIFF
transform = from_origin(x_min, y_max, cell_size, cell_size)
with rasterio.open(
    "probability_map.tif",
    "w",
    driver="GTiff",
    height=combined_layer.shape[0],
    width=combined_layer.shape[1],
    count=1,
    dtype=combined_layer.dtype,
    crs="EPSG:4326",
    transform=transform,
) as dst:
    dst.write(combined_layer, 1)

print("Probability map saved as 'probability_map.tif'")
```

```
Probability map saved as 'probability_map.tif'
```

In [19]:
```python
# Step 1: Define grid properties
width, height = 100, 100  # Grid dimensions
cell_size = 0.01  # Grid resolution in degrees
```

```
x_min, y_max = 29.0, 39.8  # Top-left corner of the grid

# Step 2: Create synthetic layers for criteria
# LKP influence (higher near the center)
x_center, y_center = 29.4, 39.5
x = np.linspace(x_min, x_min + width * cell_size, width)
y = np.linspace(y_max, y_max - height * cell_size, height)
xv, yv = np.meshgrid(x, y)

lkp_layer = np.exp(-np.sqrt((xv - x_center)**2 + (yv - y_center)**2) / 0.05)

# Settlement influence (higher near predefined points)
settlement_layer = np.zeros((height, width))
settlement_coords = [(29.3, 39.4), (29.5, 39.3)]
for sx, sy in settlement_coords:
    settlement_layer += np.exp(-np.sqrt((xv - sx)**2 + (yv - sy)**2) / 0.03)

# Terrain influence (random example values)
terrain_layer = np.random.rand(height, width)

# Step 3: Normalize layers to [0, 1]
lkp_norm = (lkp_layer - lkp_layer.min()) / (lkp_layer.max() - lkp_layer.min(
settlement_norm = (settlement_layer - settlement_layer.min()) / (settlement_
terrain_norm = (terrain_layer - terrain_layer.min()) / (terrain_layer.max()

# Step 4: Assign weights to criteria
weights = {
    "lkp": 0.5,
    "settlement": 0.3,
    "terrain": 0.2,
}

# Step 5: Compute the weighted sum
saw_result = (
    weights["lkp"] * lkp_norm +
    weights["settlement"] * settlement_norm +
    weights["terrain"] * terrain_norm
)

# Step 6: Save the SAW result as a GeoTIFF
transform = from_origin(x_min, y_max, cell_size, cell_size)
with rasterio.open(
    "saw_probability_map.tif",
    "w",
    driver="GTiff",
    height=saw_result.shape[0],
    width=saw_result.shape[1],
    count=1,
    dtype=saw_result.dtype,
    crs="EPSG:4326",
    transform=transform,
) as dst:
    dst.write(saw_result, 1)

print("SAW probability map saved as 'saw_probability_map.tif'")
```

```
SAW probability map saved as 'saw_probability_map.tif'
```

In [26]:
```
with rasterio.open("synthetic_lkp_raster.tif") as lkp_src:
    lkp_layer = lkp_src.read(1)
```

In [29]:
```
# Grid properties
width, height = 100, 100  # Grid size
cell_size = 0.01  # Cell size in degrees
x_min, y_max = 29.0, 39.8  # Top-left corner of the grid
transform = from_origin(x_min, y_max, cell_size, cell_size)

# Create synthetic raster layers
# LKP Layer (high probability near center)
x_center, y_center = 29.4, 39.5
x = np.linspace(x_min, x_min + width * cell_size, width)
y = np.linspace(y_max, y_max - height * cell_size, height)
xv, yv = np.meshgrid(x, y)
lkp_layer = np.exp(-np.sqrt((xv - x_center) ** 2 + (yv - y_center) ** 2) / 0

# Settlement Layer (higher near specific points)
settlement_layer = np.zeros((height, width))
settlement_coords = [(29.3, 39.4), (29.5, 39.3)]
for sx, sy in settlement_coords:
    settlement_layer += np.exp(-np.sqrt((xv - sx) ** 2 + (yv - sy) ** 2) / 0

# Terrain Layer (random values)
terrain_layer = np.random.rand(height, width)

# Save layers as GeoTIFF files
layers = {
    "synthetic_lkp_raster.tif": lkp_layer,
    "synthetic_settlement_raster.tif": settlement_layer,
    "synthetic_terrain_raster.tif": terrain_layer,
}

for filename, layer in layers.items():
    with rasterio.open(
        filename,
        "w",
        driver="GTiff",
        height=layer.shape[0],
        width=layer.shape[1],
        count=1,
        dtype=layer.dtype,
        crs="EPSG:4326",
        transform=transform,
    ) as dst:
        dst.write(layer, 1)

print("Synthetic raster files created:")
for filename in layers.keys():
    print(f"- {filename}")
```

```
Synthetic raster files created:
- synthetic_lkp_raster.tif
- synthetic_settlement_raster.tif
- synthetic_terrain_raster.tif
```

In [30]:
```
with rasterio.open("synthetic_settlement_raster.tif") as settlement_src:
    settlement_layer = settlement_src.read(1)

with rasterio.open("synthetic_terrain_raster.tif") as terrain_src:
    terrain_layer = terrain_src.read(1)
```

In [32]:
```
with rasterio.open("synthetic_lkp_raster.tif") as src:
    print(src.meta)
```

```
{'driver': 'GTiff', 'dtype': 'float64', 'nodata': None, 'width': 100, 'heigh
t': 100, 'count': 1, 'crs': CRS.from_wkt('GEOGCS["WGS 84",DATUM["WGS_1984",S
PHEROID["WGS 84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["
EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",
0.0174532925199433,AUTHORITY["EPSG","9122"]],AXIS["Latitude",NORTH],AXIS["Lo
ngitude",EAST],AUTHORITY["EPSG","4326"]]'), 'transform': Affine(0.01, 0.0, 2
9.0,
       0.0, -0.01, 39.8)}
```

In [34]:
```
def reproject_raster(input_path, output_path, target_crs):
    with rasterio.open(input_path) as src:
        transform, width, height = calculate_default_transform(
            src.crs, target_crs, src.width, src.height, *src.bounds
        )
        kwargs = src.meta.copy()
        kwargs.update({
            'crs': target_crs,
            'transform': transform,
            'width': width,
            'height': height
        })

        with rasterio.open(output_path, 'w', **kwargs) as dst:
            for i in range(1, src.count + 1):
                reproject(
                    source=rasterio.band(src, i),
                    destination=rasterio.band(dst, i),
                    src_transform=src.transform,
                    src_crs=src.crs,
                    dst_transform=transform,
                    dst_crs=target_crs,
                    resampling=Resampling.nearest
                )
reproject_raster("synthetic_lkp_raster.tif", "reprojected_lkp.tif", "EPSG:43
```

In [35]:
```
def normalize_raster(layer):
    return (layer - np.nanmin(layer)) / (np.nanmax(layer) - np.nanmin(layer)
```

In [36]:
```
weights = {
    "lkp": 0.5,         # LKP influence
    "settlement": 0.3,  # Settlement proximity
    "terrain": 0.2      # Terrain accessibility
}

saw_result = (
    weights["lkp"] * lkp_norm +
    weights["settlement"] * settlement_norm +
    weights["terrain"] * terrain_norm
)
```

In [38]:
```
transform = from_origin(x_min, y_max, cell_size, cell_size)
with rasterio.open(
    "saw_probability_map.tif",
    "w",
    driver="GTiff",
    height=saw_result.shape[0],
    width=saw_result.shape[1],
    count=1,
    dtype=saw_result.dtype,
    crs="EPSG:4326",
    transform=transform,
) as dst:
    dst.write(saw_result, 1)

print("SAW probability map saved as 'saw_probability_map.tif'")
```

```
SAW probability map saved as 'saw_probability_map.tif'
```

In [39]:
```python
# Load rasters
with rasterio.open("synthetic_lkp_raster.tif") as lkp_src:
    lkp_layer = lkp_src.read(1)

with rasterio.open("synthetic_settlement_raster.tif") as settlement_src:
    settlement_layer = settlement_src.read(1)

with rasterio.open("synthetic_terrain_raster.tif") as terrain_src:
    terrain_layer = terrain_src.read(1)

# Normalize layers
lkp_norm = normalize_raster(lkp_layer)
settlement_norm = normalize_raster(settlement_layer)
terrain_norm = normalize_raster(terrain_layer)

# Combine layers using SAW
weights = {"lkp": 0.5, "settlement": 0.3, "terrain": 0.2}
saw_result = (
    weights["lkp"] * lkp_norm +
    weights["settlement"] * settlement_norm +
    weights["terrain"] * terrain_norm
)

# Save the combined map
with rasterio.open(
    "saw_probability_map.tif",
    "w",
    driver="GTiff",
    height=saw_result.shape[0],
    width=saw_result.shape[1],
    count=1,
    dtype=saw_result.dtype,
    crs=lkp_src.crs,
    transform=lkp_src.transform,
) as dst:
    dst.write(saw_result, 1)

print("SAW probability map saved as 'saw_probability_map.tif'")
```

```
SAW probability map saved as 'saw_probability_map.tif'
```

In [40]:
```python
# Step 1: Load Existing Layers (LKP, Settlements, Terrain)
with rasterio.open("synthetic_lkp_raster.tif") as lkp_src:
    lkp_layer = lkp_src.read(1)
    transform = lkp_src.transform
    crs = lkp_src.crs

with rasterio.open("synthetic_settlement_raster.tif") as settlement_src:
    settlement_layer = settlement_src.read(1)

with rasterio.open("synthetic_terrain_raster.tif") as terrain_src:
    terrain_layer = terrain_src.read(1)

# Step 2: Generate Synthetic Layers for New Criteria
# Accessibility Layer (proximity to roads)
accessibility_layer = np.zeros_like(lkp_layer)
```

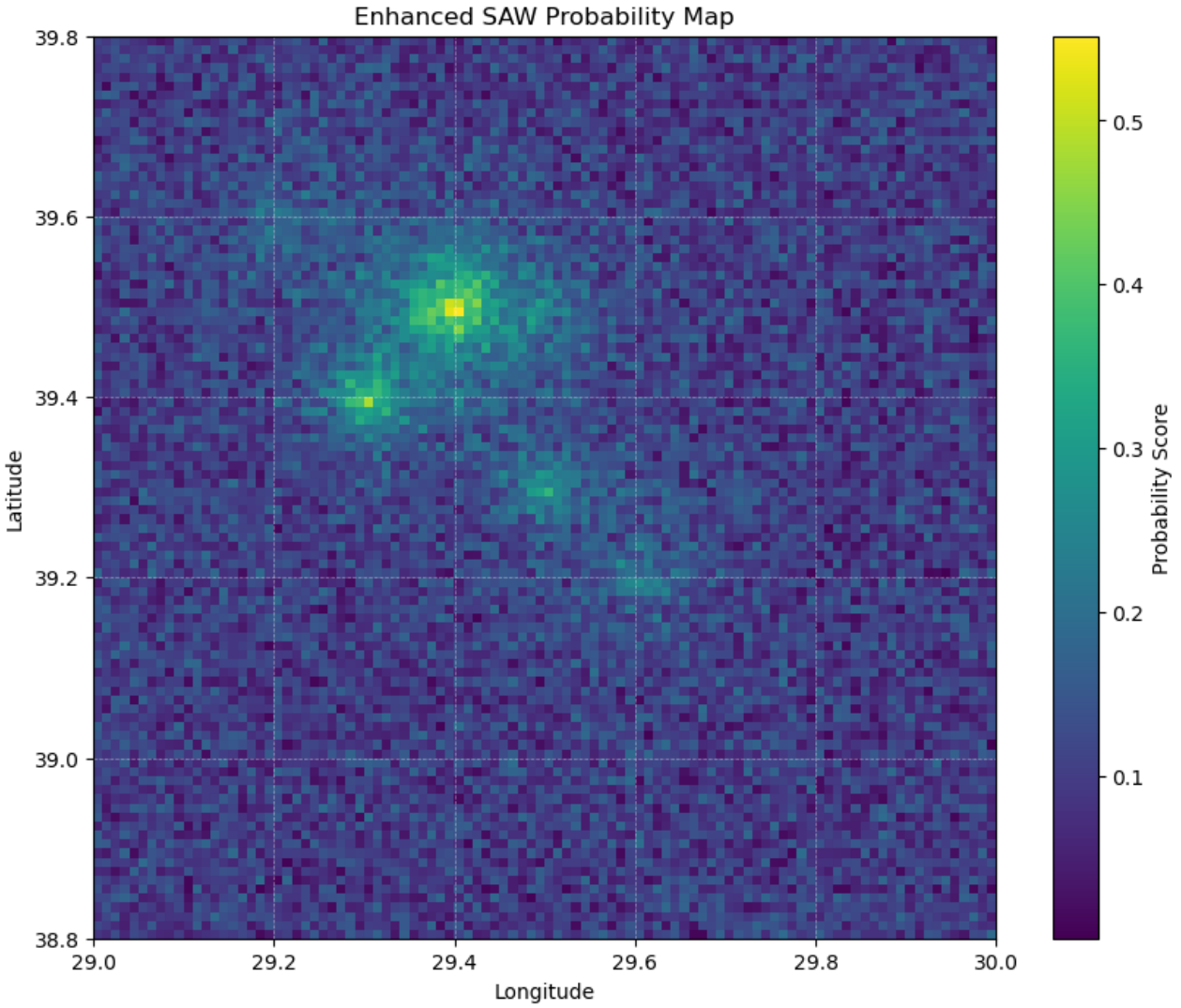
```
road_coords = [(29.3, 39.4), (29.5, 39.3)]
for rx, ry in road_coords:
    xv, yv = np.meshgrid(
        np.linspace(29.0, 30.0, accessibility_layer.shape[1]),
        np.linspace(39.0, 40.0, accessibility_layer.shape[0])
    )
    accessibility_layer += np.exp(-np.sqrt((xv - rx)**2 + (yv - ry)**2) / 0.

# Weather Layer (random suitability values)
weather_layer = np.random.rand(*lkp_layer.shape)

# Infrastructure Layer (distance to airports)
infrastructure_layer = np.zeros_like(lkp_layer)
airport_coords = [(29.2, 39.2), (29.6, 39.6)]
for ax, ay in airport_coords:
    infrastructure_layer += np.exp(-np.sqrt((xv - ax)**2 + (yv - ay)**2) / 0

# Step 3: Normalize All Layers
def normalize(layer):
    return (layer - np.nanmin(layer)) / (np.nanmax(layer) - np.nanmin(layer)

lkp_norm = normalize(lkp_layer)
settlement_norm = normalize(settlement_layer)
terrain_norm = normalize(terrain_layer)
accessibility_norm = normalize(accessibility_layer)
weather_norm = normalize(weather_layer)
infrastructure_norm = normalize(infrastructure_layer)

# Step 4: Define Weights for All Criteria
weights = {
    "lkp": 0.4,
    "settlement": 0.2,
    "terrain": 0.1,
    "accessibility": 0.1,
    "weather": 0.1,
    "infrastructure": 0.1,
}

# Step 5: Compute Weighted Overlay (SAW)
saw_result = (
    weights["lkp"] * lkp_norm +
    weights["settlement"] * settlement_norm +
    weights["terrain"] * terrain_norm +
    weights["accessibility"] * accessibility_norm +
    weights["weather"] * weather_norm +
    weights["infrastructure"] * infrastructure_norm
)

# Step 6: Save the SAW Probability Map
output_file = "enhanced_saw_probability_map.tif"
with rasterio.open(
    output_file,
    "w",
    driver="GTiff",
    height=saw_result.shape[0],
    width=saw_result.shape[1],
    count=1,
```

```
    dtype=saw_result.dtype,
    crs=crs,
    transform=transform,
) as dst:
    dst.write(saw_result, 1)

print(f"Enhanced SAW probability map saved as '{output_file}'")
```

```
Enhanced SAW probability map saved as 'enhanced_saw_probability_map.tif'
```

In [42]:

```
# Load the SAW probability map
with rasterio.open("enhanced_saw_probability_map.tif") as src:
    saw_map = src.read(1)
    extent = [src.bounds.left, src.bounds.right, src.bounds.bottom, src.boun

# Plot the probability map
plt.figure(figsize=(10, 8))
plt.imshow(saw_map, extent=extent, cmap="viridis", origin="upper")
plt.colorbar(label="Probability Score")
plt.title("Enhanced SAW Probability Map")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.grid(color='white', linestyle='--', linewidth=0.5, alpha=0.5)
plt.show()
```



In [47]:

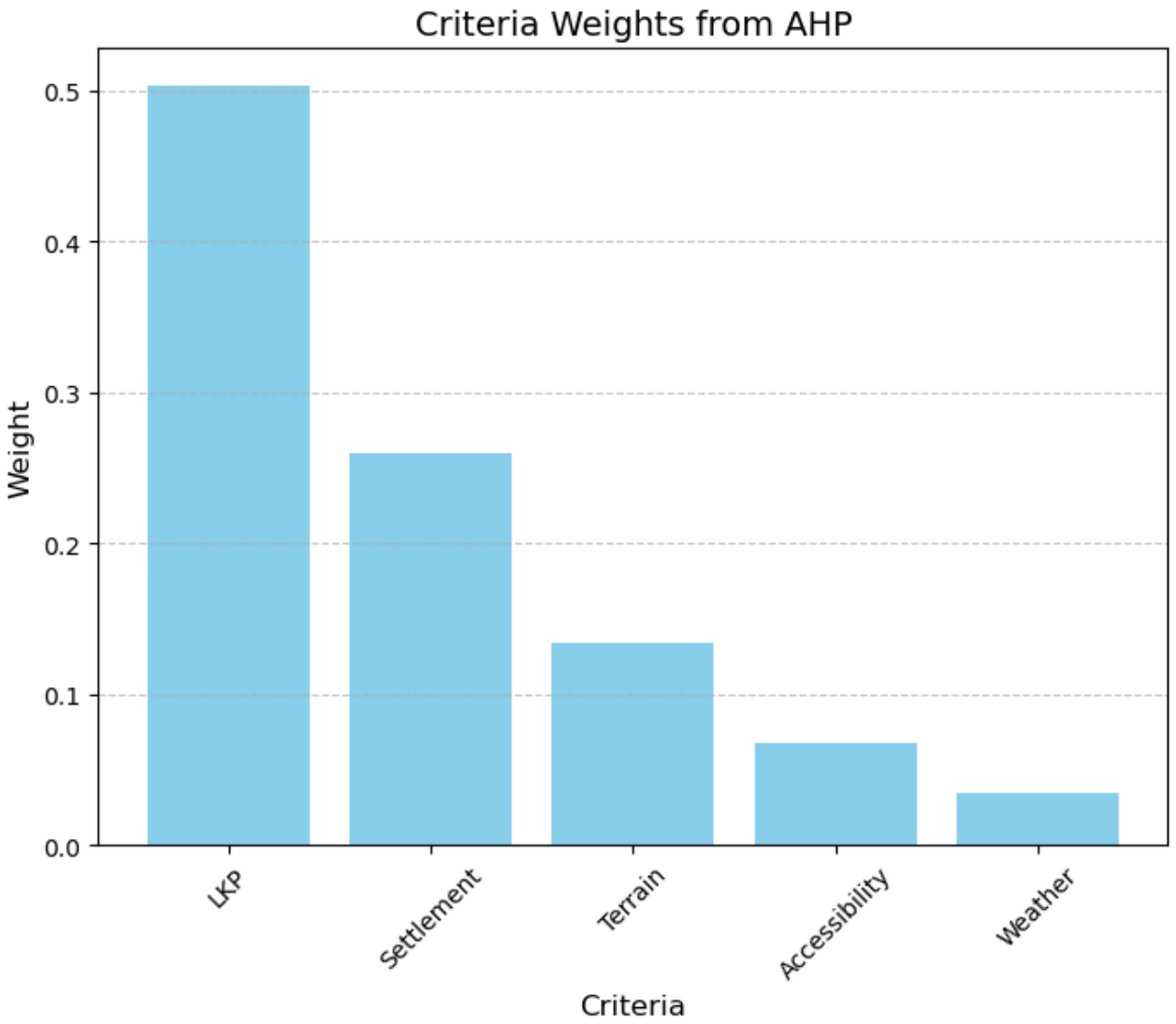
```
# Load the SAW probability map
with rasterio.open("enhanced_saw_probability_map.tif") as src:
    saw_map = src.read(1)  # Read the first band
    bounds = [[src.bounds.bottom, src.bounds.left], [src.bounds.top, src.bou

# Normalize data for Folium
saw_map_normalized = (saw_map - np.nanmin(saw_map)) / (np.nanmax(saw_map) -

# Convert the normalized array to a valid Folium input (scale to [0, 255])
saw_map_folium = (saw_map_normalized * 255).astype(np.uint8)

# Create a Folium map
m = folium.Map(
    location=[
        (bounds[0][0] + bounds[1][0]) / 2,
        (bounds[0][1] + bounds[1][1]) / 2
    ],
    zoom_start=10
)

# Overlay the probability map
folium.raster_layers.ImageOverlay(
    image=saw_map_folium,
    bounds=bounds,
    colormap=lambda x: (1, 0, 0, x),  # Red with transparency
    name="SAW Probability Map"
).add_to(m)

# Add layer control and save the map
folium.LayerControl().add_to(m)
m.save("saw_probability_map.html")
print("Map saved as 'saw_probability_map.html'")
```

```
Map saved as 'saw_probability_map.html'
```

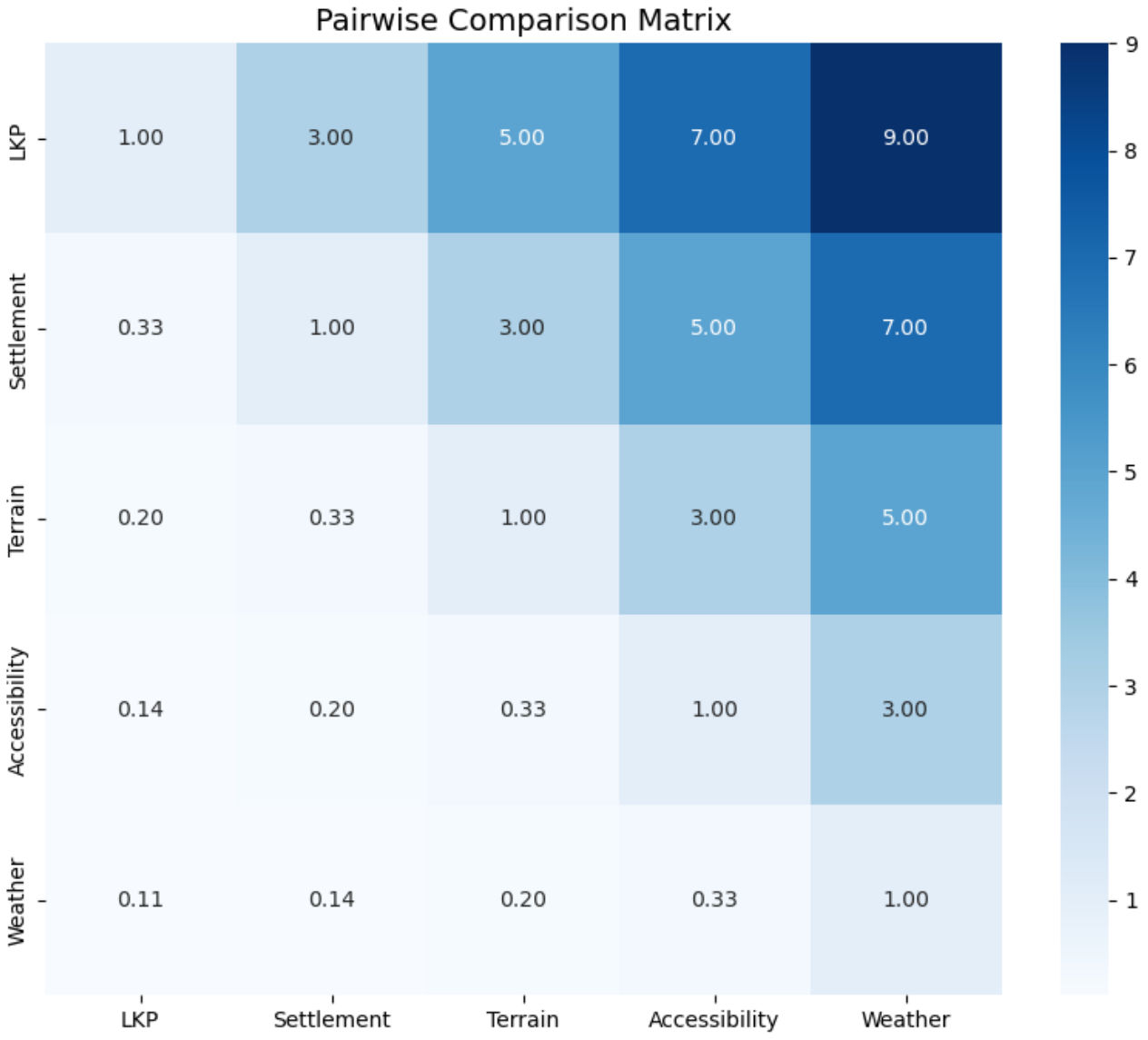
In [51]:
```
# Step 1: Define Criteria and Pairwise Comparison Matrix
criteria = ["LKP", "Settlement", "Terrain", "Accessibility", "Weather"]
pairwise_matrix = np.array([
    [1, 3, 5, 7, 9],  # LKP compared to others
    [1/3, 1, 3, 5, 7],  # Settlement compared to others
    [1/5, 1/3, 1, 3, 5],  # Terrain compared to others
    [1/7, 1/5, 1/3, 1, 3],  # Accessibility compared to others
    [1/9, 1/7, 1/5, 1/3, 1]  # Weather compared to others
])

# Step 2: Normalize the Pairwise Matrix
column_sums = pairwise_matrix.sum(axis=0)
normalized_matrix = pairwise_matrix / column_sums

# Step 3: Calculate Criteria Weights
criteria_weights = normalized_matrix.mean(axis=1)

# Step 4: Display the Results
weights_df = pd.DataFrame({
    "Criteria": criteria,
    "Weight": criteria_weights
}).sort_values(by="Weight", ascending=False)

print("Criteria Weights:")
print(weights_df)

# Step 5: Check Consistency
# Calculate Consistency Index (CI) and Consistency Ratio (CR)
eigenvalues = np.dot(pairwise_matrix, criteria_weights) / criteria_weights
lambda_max = eigenvalues.mean()
n = pairwise_matrix.shape[0]
ci = (lambda_max - n) / (n - 1)
random_index = {1: 0.00, 2: 0.00, 3: 0.58, 4: 0.90, 5: 1.12, 6: 1.24, 7: 1.3
cr = ci / random_index[n]

print(f"\nConsistency Index (CI): {ci:.4f}")
print(f"Consistency Ratio (CR): {cr:.4f}")

if cr < 0.1:
    print("The pairwise comparisons are consistent.")
else:
    print("The pairwise comparisons are not consistent. Reassess the matrix.
```

```
Criteria Weights:
        Criteria    Weight
0            LKP  0.502819
1     Settlement  0.260232
2        Terrain  0.134350
3  Accessibility  0.067778
4        Weather  0.034821

Consistency Index (CI): 0.0607
Consistency Ratio (CR): 0.0542
The pairwise comparisons are consistent.
```

In [52]:
```
# Visualization: Criteria Weights
plt.figure(figsize=(8, 6))
plt.bar(weights_df["Criteria"], weights_df["Weight"], color="skyblue")
plt.title("Criteria Weights from AHP", fontsize=14)
plt.xlabel("Criteria", fontsize=12)
plt.ylabel("Weight", fontsize=12)
plt.xticks(rotation=45)
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```

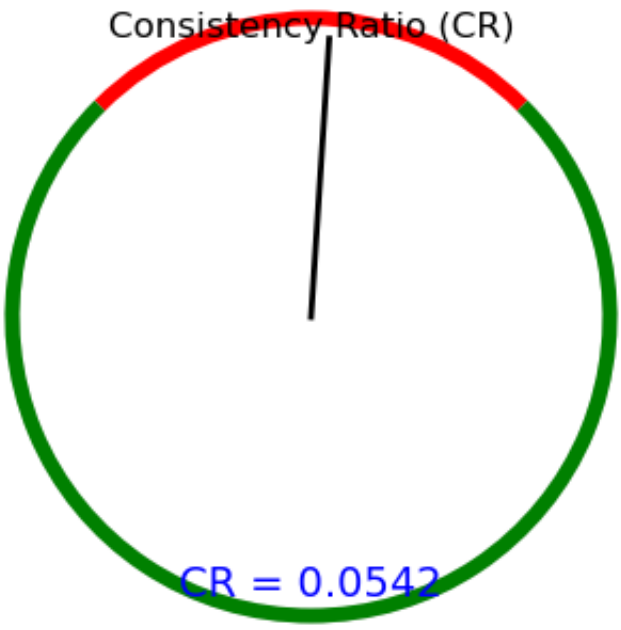


In [54]:
```
# Convert Pairwise Matrix to DataFrame for Visualization
pairwise_df = pd.DataFrame(pairwise_matrix, index=criteria, columns=criteria

# Visualization: Heatmap of Pairwise Comparisons
plt.figure(figsize=(10, 8))
sns.heatmap(pairwise_df, annot=True, fmt=".2f", cmap="Blues", cbar=True)
plt.title("Pairwise Comparison Matrix", fontsize=14)
plt.show()
```

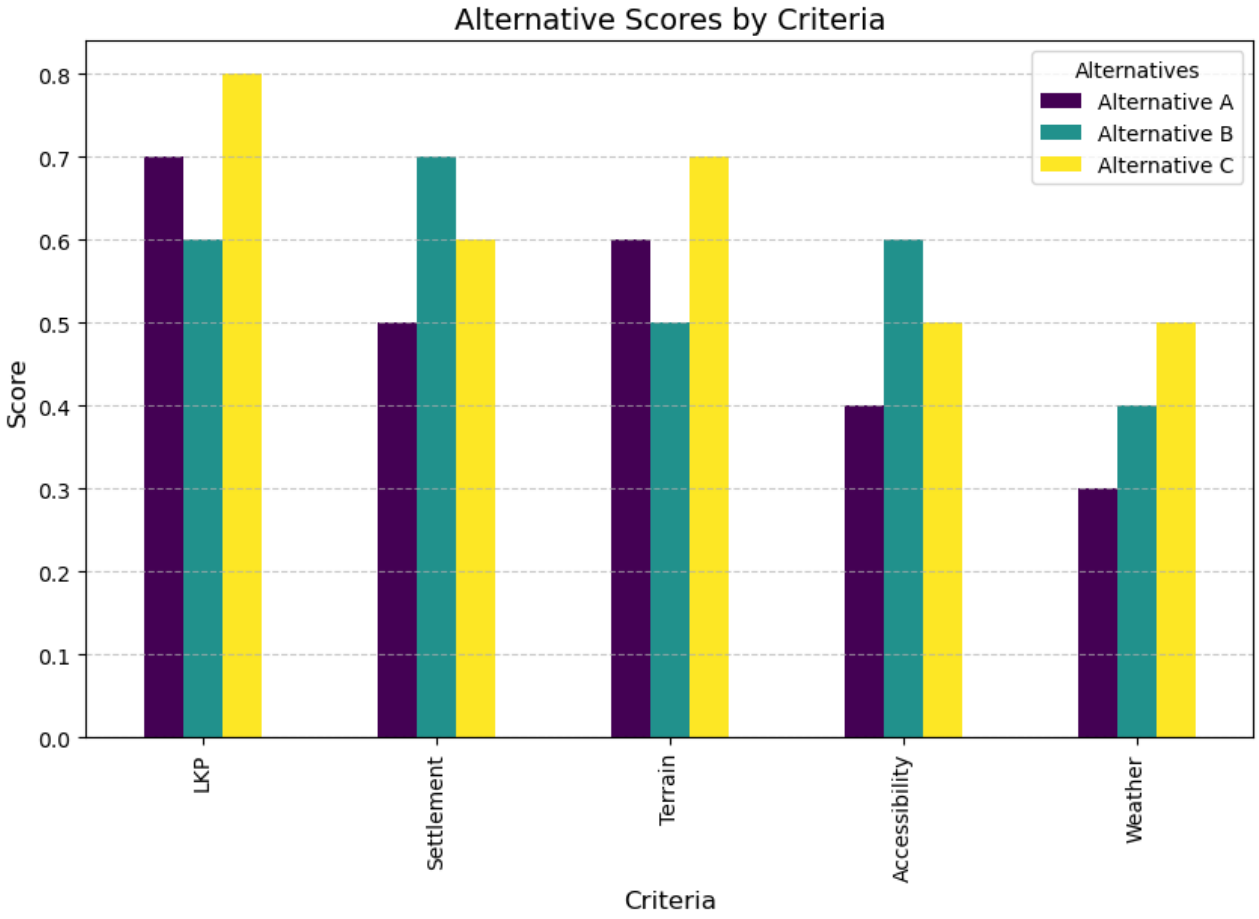

Pairwise Comparison Matrix
LKP
Settlement
Terrain
Accessibility
Weather
1.00
3.00
5.00
7.00
9.00
0.33
1.00
3.00
5.00
7.00
0.20
0.33
1.00
3.00
5.00
0.14
0.20
0.33
1.00
3.00
0.11
0.14
0.20
0.33
1.00
LKP
Settlement
Terrain
Accessibility
Weather
9
8
7
6
5
4
3
2
1

In [56]:

```
# Function to Draw a Gauge
def plot_gauge(cr, threshold=0.1):
    fig, ax = plt.subplots(figsize=(6, 4), subplot_kw={"aspect": "equal"})

    # Draw the base arc
    theta1, theta2 = 135, 45
    ax.add_patch(patches.Arc((0.5, 0.5), 1.5, 1.5, theta1=theta1, theta2=the

    # Threshold marker
    angle_threshold = 135 - (threshold / 0.1 * 90)
    ax.add_patch(patches.Arc((0.5, 0.5), 1.5, 1.5, theta1=angle_threshold, t
    ax.add_patch(patches.Arc((0.5, 0.5), 1.5, 1.5, theta1=135, theta2=angle_

    # CR marker
    angle_cr = 135 - (cr / 0.1 * 90)
    ax.plot([0.5, 0.5 + 0.7 * np.cos(np.radians(angle_cr))],
            [0.5, 0.5 + 0.7 * np.sin(np.radians(angle_cr))],
            color="black", lw=2)

    # Annotations
    ax.text(0.5, 1.2, "Consistency Ratio (CR)", ha="center", fontsize=12)
    ax.text(0.5, -0.2, f"CR = {cr:.4f}", ha="center", fontsize=14, color="bl
    ax.axis("off")
    plt.show()

# Visualize CR Gauge
plot_gauge(cr)
```



In [57]:
```
# Example Data (Weights for 3 Alternatives)
alternative_scores = pd.DataFrame({
    "Criteria": ["LKP", "Settlement", "Terrain", "Accessibility", "Weather"]
    "Alternative A": [0.7, 0.5, 0.6, 0.4, 0.3],
    "Alternative B": [0.6, 0.7, 0.5, 0.6, 0.4],
    "Alternative C": [0.8, 0.6, 0.7, 0.5, 0.5],
}).set_index("Criteria")

# Plot
alternative_scores.plot(kind="bar", figsize=(10, 6), colormap="viridis")
plt.title("Alternative Scores by Criteria", fontsize=14)
plt.ylabel("Score", fontsize=12)
plt.xlabel("Criteria", fontsize=12)
plt.legend(title="Alternatives")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



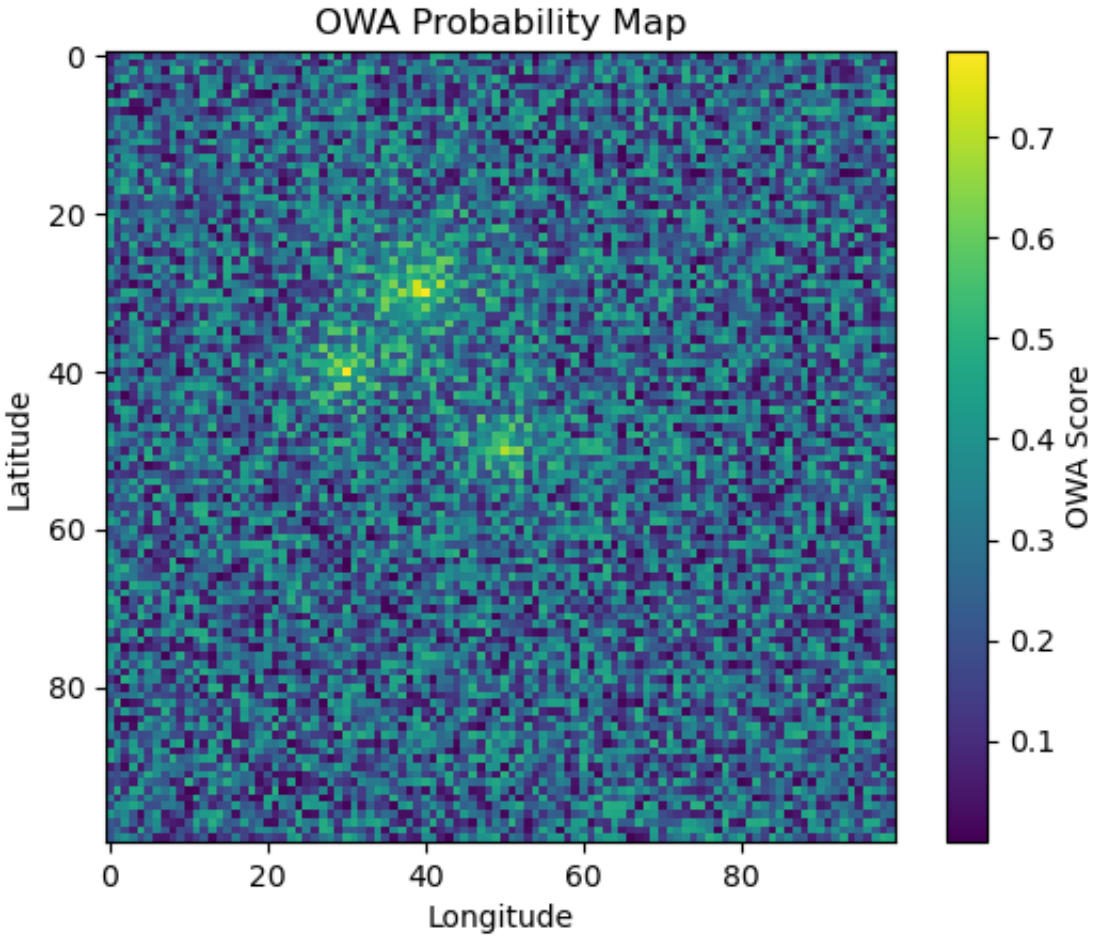
In [58]:
```python
# Step 1: Define Criteria Values for an Example Alternative
# Example: Normalized values for one grid cell or region
criteria = ["LKP", "Settlement", "Terrain", "Accessibility", "Weather"]
values = np.array([0.9, 0.7, 0.5, 0.3, 0.2])  # Normalized values (example)

# Step 2: Sort Criteria Values in Descending Order
sorted_values = np.sort(values)[::-1]  # Descending order

# Step 3: Define OWA Weights
# Example: Neutral aggregation (equal weights)
owa_weights = np.array([0.2, 0.2, 0.2, 0.2, 0.2])  # Must sum to 1

# Adjust OWA weights for optimism or pessimism
# Optimistic: [1, 0, 0, 0, 0] (focus on the highest value)
# Pessimistic: [0, 0, 0, 0, 1] (focus on the lowest value)

# Step 4: Compute OWA Score
owa_score = np.sum(owa_weights * sorted_values)

# Display Results
print(f"Criteria: {criteria}")
print(f"Original Values: {values}")
print(f"Sorted Values: {sorted_values}")
print(f"OWA Weights: {owa_weights}")
print(f"OWA Score: {owa_score:.4f}")
```

```
Criteria: ['LKP', 'Settlement', 'Terrain', 'Accessibility', 'Weather']
Original Values: [0.9 0.7 0.5 0.3 0.2]
Sorted Values: [0.9 0.7 0.5 0.3 0.2]
OWA Weights: [0.2 0.2 0.2 0.2 0.2]
OWA Score: 0.5200
```

In [59]:
```
# Load Criteria Layers (Rasters)
with rasterio.open("synthetic_lkp_raster.tif") as lkp_src:
    lkp_layer = lkp_src.read(1)  # Read the first band

with rasterio.open("synthetic_settlement_raster.tif") as settlement_src:
    settlement_layer = settlement_src.read(1)

with rasterio.open("synthetic_terrain_raster.tif") as terrain_src:
    terrain_layer = terrain_src.read(1)

# Combine Layers into a 3D Array (Criteria Layers)
criteria_layers = np.stack([lkp_layer, settlement_layer, terrain_layer], axi

# Normalize Layers to [0, 1]
criteria_layers = (criteria_layers - criteria_layers.min(axis=(1, 2), keepdi
    criteria_layers.max(axis=(1, 2), keepdims=True) - criteria_layers.min(ax
)

# Define OWA Weights (Neutral Example)
owa_weights = np.array([0.5, 0.3, 0.2])  # Adjust based on strategy (must su

# Compute OWA Score for Each Cell
owa_scores = np.zeros((criteria_layers.shape[1], criteria_layers.shape[2]))
for i in range(criteria_layers.shape[1]):
    for j in range(criteria_layers.shape[2]):
        sorted_values = np.sort(criteria_layers[:, i, j])[::-1]
        owa_scores[i, j] = np.sum(owa_weights * sorted_values)

# Save OWA Scores as GeoTIFF
with rasterio.open(
    "owa_probability_map.tif",
    "w",
    driver="GTiff",
    height=owa_scores.shape[0],
    width=owa_scores.shape[1],
    count=1,
    dtype=owa_scores.dtype,
    crs=lkp_src.crs,
    transform=lkp_src.transform,
) as dst:
    dst.write(owa_scores, 1)

print("OWA probability map saved as 'owa_probability_map.tif'")
```

```
OWA probability map saved as 'owa_probability_map.tif'
```

In [60]:
```
plt.imshow(owa_scores, cmap="viridis", origin="upper")
plt.colorbar(label="OWA Score")
plt.title("OWA Probability Map")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```


OWA Probability Map
Latitude
Longitude
OWA Score
0
20
40
60
80
0.1
0.2
0.3
0.4
0.5
0.6
0.7

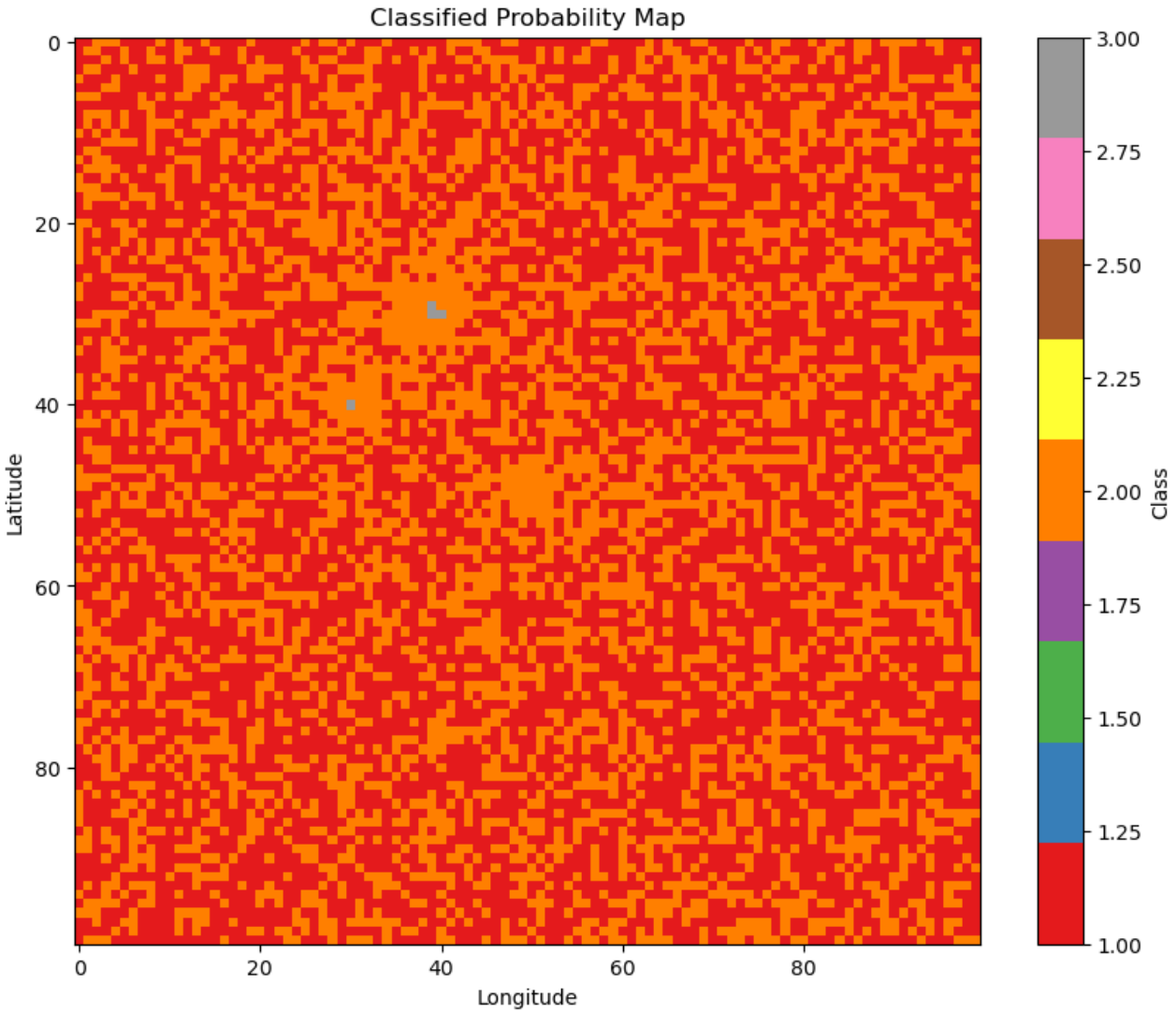
In [61]:

```
# Load the OWA Probability Map
with rasterio.open("owa_probability_map.tif") as src:
    owa_map = src.read(1)  # Read the first band
    bounds = [[src.bounds.bottom, src.bounds.left], [src.bounds.top, src.bou

# Normalize the OWA map for display in Folium
owa_map_normalized = (owa_map - np.nanmin(owa_map)) / (np.nanmax(owa_map) -
owa_map_normalized = (owa_map_normalized * 255).astype(np.uint8)  # Scale to

# Create a Folium Map
m = folium.Map(
    location=[(bounds[0][0] + bounds[1][0]) / 2, (bounds[0][1] + bounds[1][1
    zoom_start=10,
    tiles="cartodbpositron"
)

# Overlay the OWA map
folium.raster_layers.ImageOverlay(
    image=owa_map_normalized,
    bounds=bounds,
    colormap=lambda x: (x, 0, 1 - x, 0.6),  # Blue gradient with transparenc
    name="OWA Probability Map"
).add_to(m)

# Add a layer control and save the map
folium.LayerControl().add_to(m)
m.save("owa_probability_map.html")
print("Map saved as 'owa_probability_map.html'")
```

```
Map saved as 'owa_probability_map.html'
```

In [63]:

```
# Step 1: Load the Probability Map
with rasterio.open("owa_probability_map.tif") as src:
    probability_map = src.read(1)  # Read the first band
    transform = src.transform
    crs = src.crs

# Step 2: Define Classification Thresholds
# Example: Custom thresholds for 3 classes (Low, Medium, High)
thresholds = [0.0, 0.3, 0.7, 1.0]  # Adjust as needed

# Step 3: Classify the Map
classified_map = np.digitize(probability_map, bins=thresholds, right=True)

# Step 4: Visualize the Classified Map
plt.figure(figsize=(10, 8))
plt.imshow(classified_map, cmap="Set1", origin="upper")
plt.colorbar(label="Class")
plt.title("Classified Probability Map")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()

# Step 5: Save the Classified Map
output_file = "classified_probability_map.tif"
with rasterio.open(
    output_file,
    "w",
    driver="GTiff",
    height=classified_map.shape[0],
    width=classified_map.shape[1],
    count=1,
    dtype=classified_map.dtype,
    crs=crs,
    transform=transform,
) as dst:
    dst.write(classified_map, 1)

print(f"Classified probability map saved as '{output_file}'")
```




```
Classified probability map saved as 'classified_probability_map.tif'
```

In [64]:
```
thresholds = np.linspace(np.min(probability_map), np.max(probability_map), n
```

In [65]:
```
thresholds = np.percentile(probability_map[~np.isnan(probability_map)], [0,
```

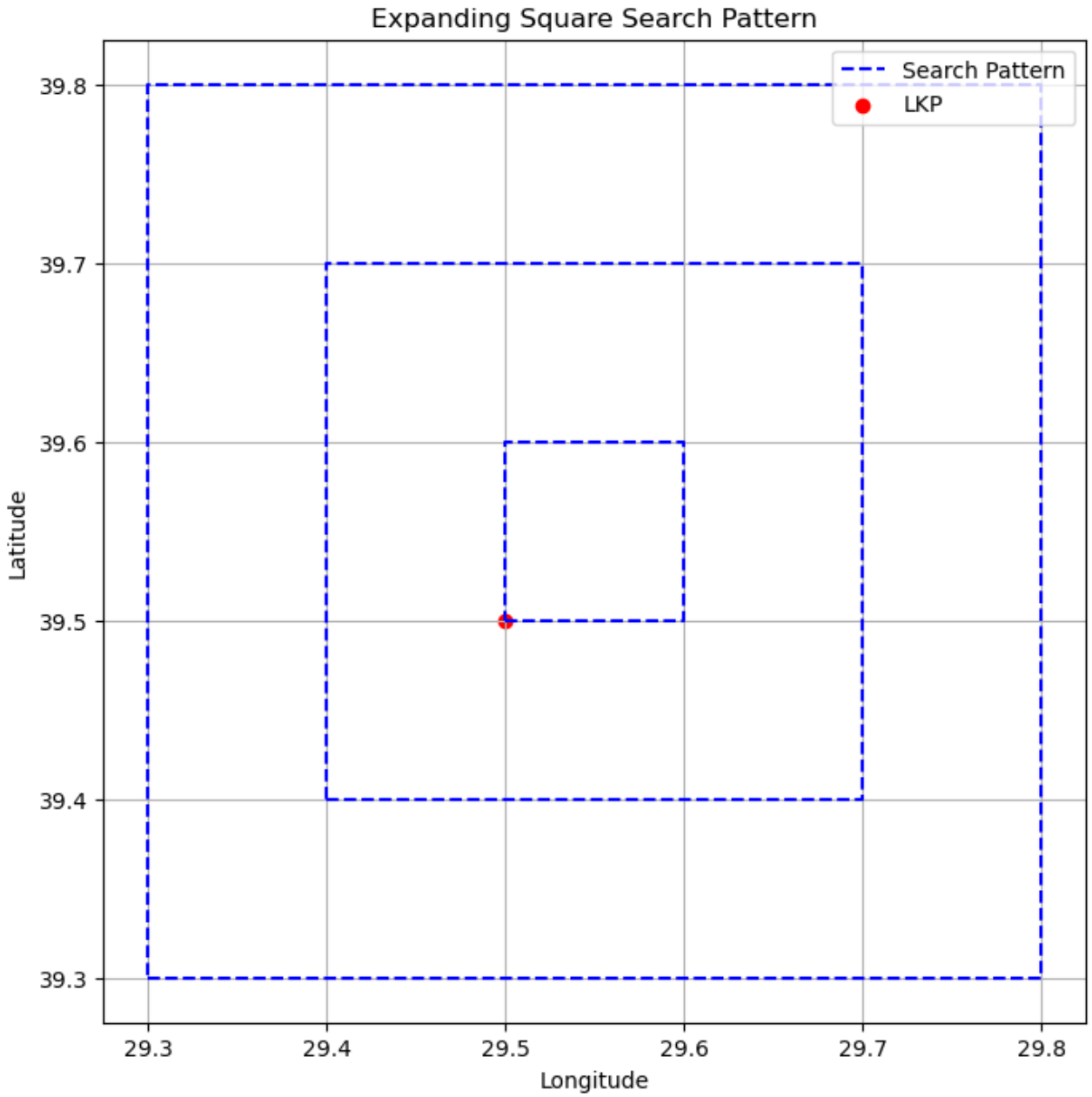
In [71]:
```
# Flatten the probability map, excluding NaN values
data = probability_map[~np.isnan(probability_map)].flatten()

# Specify the number of classes
num_classes = 3

# Calculate Jenks natural breaks
thresholds = jenkspy.jenks_breaks(data, num_classes)

print("Jenks Natural Breaks Thresholds:", thresholds)
```

```
Jenks Natural Breaks Thresholds: [0.00014358225047216507, 0.1766282378287802
6, 0.3456137984942614, 0.7852180498304305]
```

In [72]:
```
classified_map = np.digitize(probability_map, bins=thresholds, right=True)
```

In [73]:

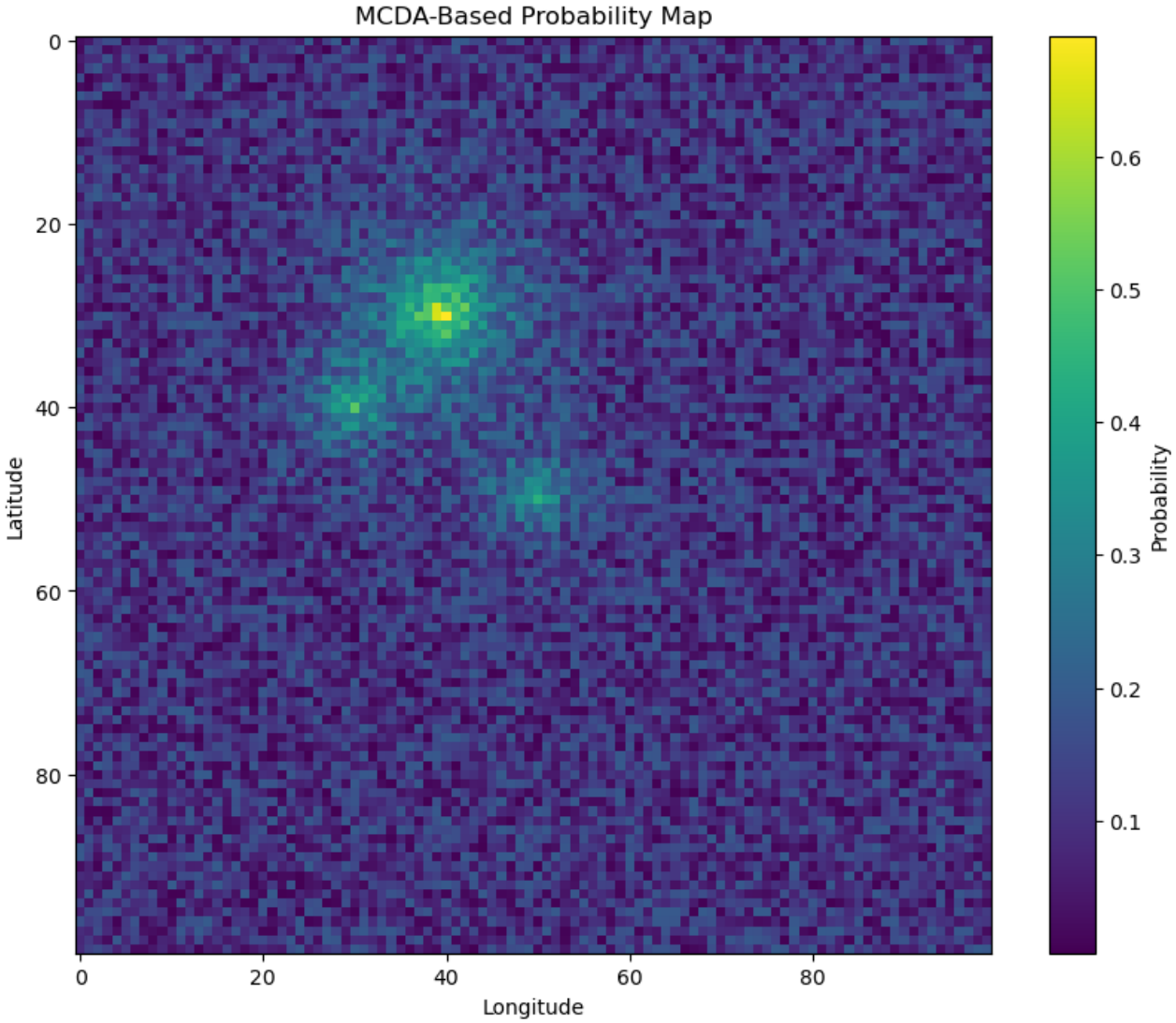
```
import matplotlib.pyplot as plt
from shapely.geometry import Point, LineString, Polygon
import geopandas as gpd

# Define the Last Known Position (LKP)
lkp = Point(29.5, 39.5)

# Create Expanding Square Search Pattern
square_coords = [
    [(29.5, 39.5), (29.6, 39.5), (29.6, 39.6), (29.5, 39.6), (29.5, 39.5)],
    [(29.4, 39.4), (29.7, 39.4), (29.7, 39.7), (29.4, 39.7), (29.4, 39.4)],
    [(29.3, 39.3), (29.8, 39.3), (29.8, 39.8), (29.3, 39.8), (29.3, 39.3)],
]

# Convert to GeoDataFrame
squares = [Polygon(coords) for coords in square_coords]
search_pattern_gdf = gpd.GeoDataFrame(geometry=squares, crs="EPSG:4326")

# Plot the Search Pattern
fig, ax = plt.subplots(figsize=(10, 8))
search_pattern_gdf.boundary.plot(ax=ax, color="blue", linestyle="--", label=
gpd.GeoSeries([lkp]).plot(ax=ax, color="red", label="LKP")
plt.title("Expanding Square Search Pattern")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend()
plt.grid()
plt.show()
```



In [74]:
```python
# Step 1: Load Criteria Layers
with rasterio.open("synthetic_lkp_raster.tif") as lkp_src:
    lkp_layer = lkp_src.read(1)

with rasterio.open("synthetic_settlement_raster.tif") as settlement_src:
    settlement_layer = settlement_src.read(1)

with rasterio.open("synthetic_terrain_raster.tif") as terrain_src:
    terrain_layer = terrain_src.read(1)

# Step 2: Normalize Criteria Layers (0 to 1 scale)
def normalize(layer):
    return (layer - np.nanmin(layer)) / (np.nanmax(layer) - np.nanmin(layer)

lkp_norm = normalize(lkp_layer)
settlement_norm = normalize(settlement_layer)
```

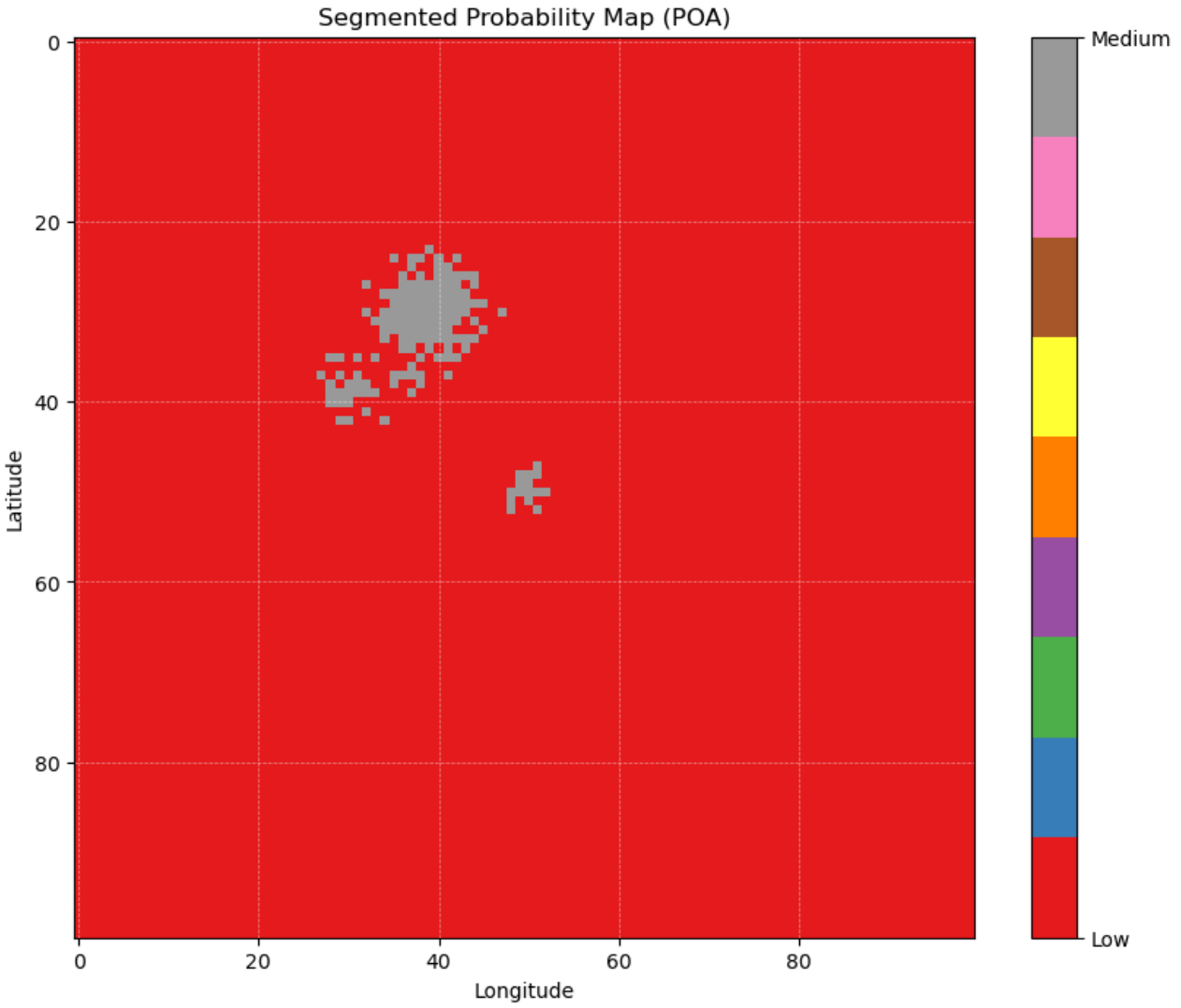
```
terrain_norm = normalize(terrain_layer)

# Step 3: Define Weights for Each Criterion
weights = {
    "lkp": 0.5,         # LKP Proximity
    "settlement": 0.3,  # Settlement Proximity
    "terrain": 0.2      # Terrain Accessibility
}

# Step 4: Compute Weighted Sum
probability_map = (
    weights["lkp"] * lkp_norm +
    weights["settlement"] * settlement_norm +
    weights["terrain"] * terrain_norm
)

# Step 5: Save Probability Map as GeoTIFF
transform = lkp_src.transform
crs = lkp_src.crs
with rasterio.open(
    "mcda_probability_map.tif",
    "w",
    driver="GTiff",
    height=probability_map.shape[0],
    width=probability_map.shape[1],
    count=1,
    dtype=probability_map.dtype,
    crs=crs,
    transform=transform,
) as dst:
    dst.write(probability_map, 1)

print("MCDA-based probability map saved as 'mcda_probability_map.tif'")

# Step 6: Visualize Probability Map
plt.figure(figsize=(10, 8))
plt.imshow(probability_map, cmap="viridis", origin="upper")
plt.colorbar(label="Probability")
plt.title("MCDA-Based Probability Map")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```

```
MCDA-based probability map saved as 'mcda_probability_map.tif'
```


MCDA-Based Probability Map
Latitude
Longitude
Probability
0
20
40
60
80
0.1
0.2
0.3
0.4
0.5
0.6

In [75]:

```
# Step 1: Load Probability Map
with rasterio.open("mcda_probability_map.tif") as src:
    probability_map = src.read(1)
    transform = src.transform
    crs = src.crs

# Step 2: Define Thresholds
# Define thresholds for Low, Medium, High
thresholds = [0.0, 0.3, 0.7, 1.0]  # Low: 0-0.3, Medium: 0.3-0.7, High: 0.7-
labels = [1, 2, 3]  # 1: Low, 2: Medium, 3: High

# Step 3: Segment Probability Map
segmented_map = np.digitize(probability_map, bins=thresholds, right=True)

# Step 4: Visualize Segmented Map
plt.figure(figsize=(10, 8))
plt.imshow(segmented_map, cmap="Set1", origin="upper")
cbar = plt.colorbar(ticks=[1, 2, 3])
cbar.ax.set_yticklabels(["Low", "Medium", "High"])
plt.title("Segmented Probability Map (POA)")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.grid(color="white", linestyle="--", linewidth=0.5, alpha=0.5)
plt.show()

# Step 5: Save Segmented Map as GeoTIFF
output_file = "segmented_probability_map.tif"
with rasterio.open(
    output_file,
    "w",
    driver="GTiff",
    height=segmented_map.shape[0],
    width=segmented_map.shape[1],
    count=1,
    dtype=segmented_map.dtype,
    crs=crs,
    transform=transform,
) as dst:
    dst.write(segmented_map, 1)

print(f"Segmented probability map saved as '{output_file}'")
```



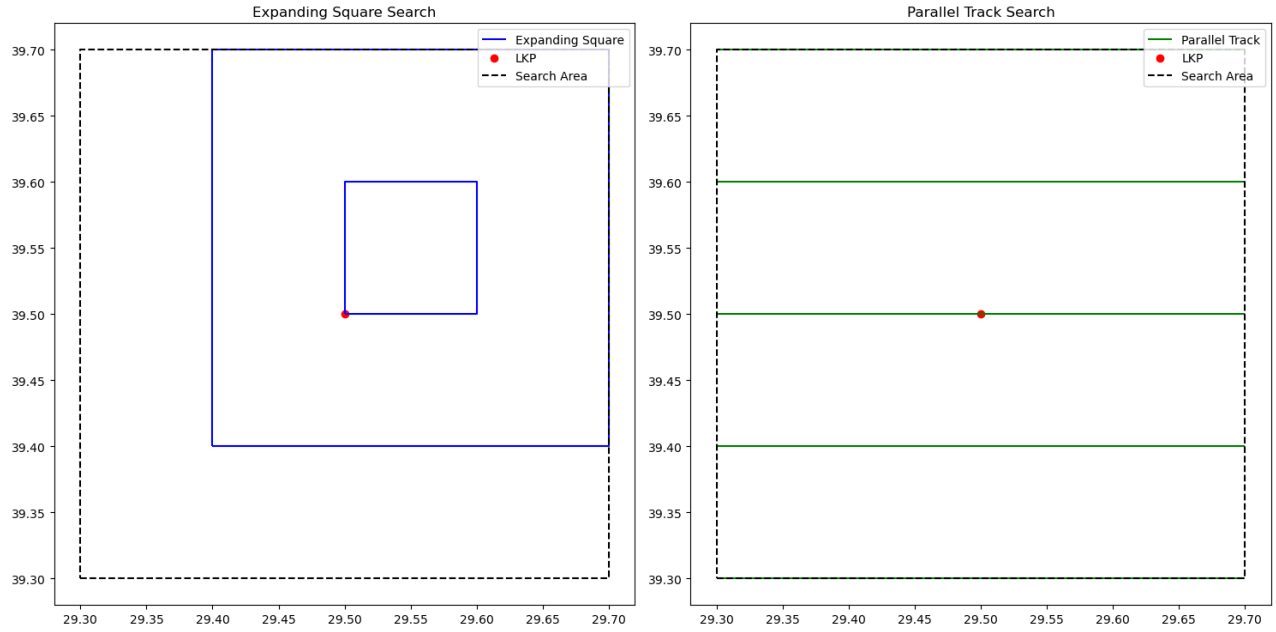
Segmented probability map saved as 'segmented_probability_map.tif'

In [76]:
```
# Step 1: Count the number of cells in each segment
unique, counts = np.unique(segmented_map, return_counts=True)
cell_counts = dict(zip(unique, counts))

# Step 2: Calculate area (assuming square grid cells)
cell_area = (transform[0] ** 2)  # Cell size in map units (e.g., degrees or
area_by_class = {label: count * cell_area for label, count in cell_counts.it

# Display results
for label, area in area_by_class.items():
    class_name = ["Low", "Medium", "High"][label - 1]
    print(f"Class '{class_name}' covers an area of {area:.2f} square units."
```

```
Class 'Low' covers an area of 0.99 square units.
Class 'Medium' covers an area of 0.01 square units.
```

In [78]:

```
# Define Search Area and LKP
lkp = Point(29.5, 39.5)
search_area = Polygon([(29.3, 39.3), (29.7, 39.3), (29.7, 39.7), (29.3, 39.7

# Generate Search Patterns
# 1. Expanding Square
expanding_square_coords = [
    [(29.5, 39.5), (29.6, 39.5), (29.6, 39.6), (29.5, 39.6), (29.5, 39.5)],
    [(29.4, 39.4), (29.7, 39.4), (29.7, 39.7), (29.4, 39.7), (29.4, 39.4)]
]
expanding_square = [Polygon(coords) for coords in expanding_square_coords]

# 2. Parallel Track
parallel_tracks = [
    LineString([(29.3, y), (29.7, y)]) for y in [39.3, 39.4, 39.5, 39.6, 39.
]

# Convert to GeoDataFrames
expanding_square_gdf = gpd.GeoDataFrame(geometry=expanding_square, crs="EPSG
parallel_track_gdf = gpd.GeoDataFrame(geometry=parallel_tracks, crs="EPSG:43

# Visualize the Search Patterns
fig, ax = plt.subplots(1, 2, figsize=(15, 8))

# Expanding Square Pattern
expanding_square_gdf.boundary.plot(ax=ax[0], color="blue", label="Expanding
gpd.GeoSeries([lkp]).plot(ax=ax[0], color="red", label="LKP")
gpd.GeoSeries([search_area]).boundary.plot(ax=ax[0], color="black", linestyl
ax[0].set_title("Expanding Square Search")
ax[0].legend()

# Parallel Track Pattern
parallel_track_gdf.plot(ax=ax[1], color="green", label="Parallel Track")
gpd.GeoSeries([lkp]).plot(ax=ax[1], color="red", label="LKP")
gpd.GeoSeries([search_area]).boundary.plot(ax=ax[1], color="black", linestyl
ax[1].set_title("Parallel Track Search")
ax[1].legend()

plt.tight_layout()
plt.show()
```



In [81]:
```
# Define coordinates for the high-probability area (example)
high_prob_coords = [
    (29.4, 39.4), (29.6, 39.4), (29.6, 39.6), (29.4, 39.6), (29.4, 39.4)
]

# Create the high-probability polygon
high_prob_area = gpd.GeoDataFrame(geometry=[Polygon(high_prob_coords)], crs=
```

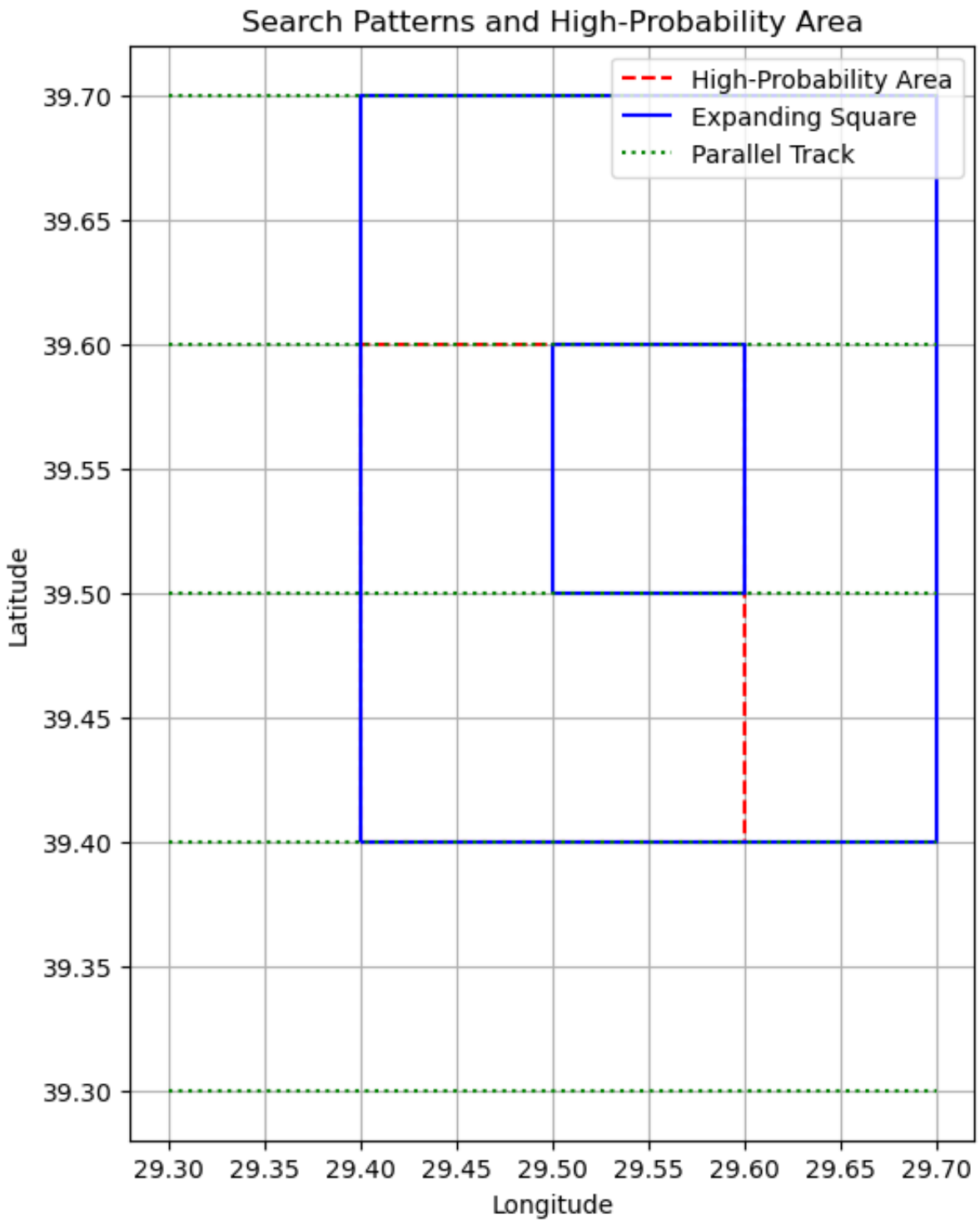
In [82]:
```
# Calculate the coverage of high-probability areas
expanding_square_coverage = expanding_square_gdf.intersection(high_prob_area
parallel_track_coverage = parallel_track_gdf.intersection(high_prob_area.una

# Display the results
print(f"Expanding Square Coverage (area units): {expanding_square_coverage}"
print(f"Parallel Track Coverage (area units): {parallel_track_coverage}")
```

```
/var/folders/fp/drm5pw1d13x1n4wvd2q14v0m0000gn/T/ipykernel_26763/3804832417.
py:2: DeprecationWarning: The 'unary_union' attribute is deprecated, use the
'union_all()' method instead.
  expanding_square_coverage = expanding_square_gdf.intersection(high_prob_ar
ea.unary_union).area.sum()
/var/folders/fp/drm5pw1d13x1n4wvd2q14v0m0000gn/T/ipykernel_26763/3804832417.
py:2: UserWarning: Geometry is in a geographic CRS. Results from 'area' are
likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a pro
jected CRS before this operation.

  expanding_square_coverage = expanding_square_gdf.intersection(high_prob_ar
ea.unary_union).area.sum()
/var/folders/fp/drm5pw1d13x1n4wvd2q14v0m0000gn/T/ipykernel_26763/3804832417.
py:3: DeprecationWarning: The 'unary_union' attribute is deprecated, use the
'union_all()' method instead.
  parallel_track_coverage = parallel_track_gdf.intersection(high_prob_area.u
nary_union).area.sum()
Expanding Square Coverage (area units): 0.050000000000001425
Parallel Track Coverage (area units): 0.0
```

```
/var/folders/fp/drm5pw1d13x1n4wvd2q14v0m0000gn/T/ipykernel_26763/3804832417.
py:3: UserWarning: Geometry is in a geographic CRS. Results from 'area' are
likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a pro
jected CRS before this operation.

  parallel_track_coverage = parallel_track_gdf.intersection(high_prob_area.u
nary_union).area.sum()
```

In [83]:

```
# Visualize Search Patterns and High-Probability Area
fig, ax = plt.subplots(figsize=(10, 8))

high_prob_area.boundary.plot(ax=ax, color="red", linestyle="--", label="High
expanding_square_gdf.boundary.plot(ax=ax, color="blue", label="Expanding Squ
parallel_track_gdf.plot(ax=ax, color="green", linestyle=":", label="Parallel

plt.title("Search Patterns and High-Probability Area")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.legend()
plt.grid()
plt.show()
```



In [86]:
```
# Streamlit UI Setup
st.title("Search Pattern Comparison and Probability Mapping")
st.sidebar.header("Settings")

# Step 1: Define the High-Probability Area
st.sidebar.subheader("High-Probability Area")
x_min = st.sidebar.slider("X Min", 29.0, 30.0, 29.4)
x_max = st.sidebar.slider("X Max", 29.0, 30.0, 29.6)
y_min = st.sidebar.slider("Y Min", 39.0, 40.0, 39.4)
y_max = st.sidebar.slider("Y Max", 39.0, 40.0, 39.6)

high_prob_area_coords = [(x_min, y_min), (x_max, y_min), (x_max, y_max), (x_
```

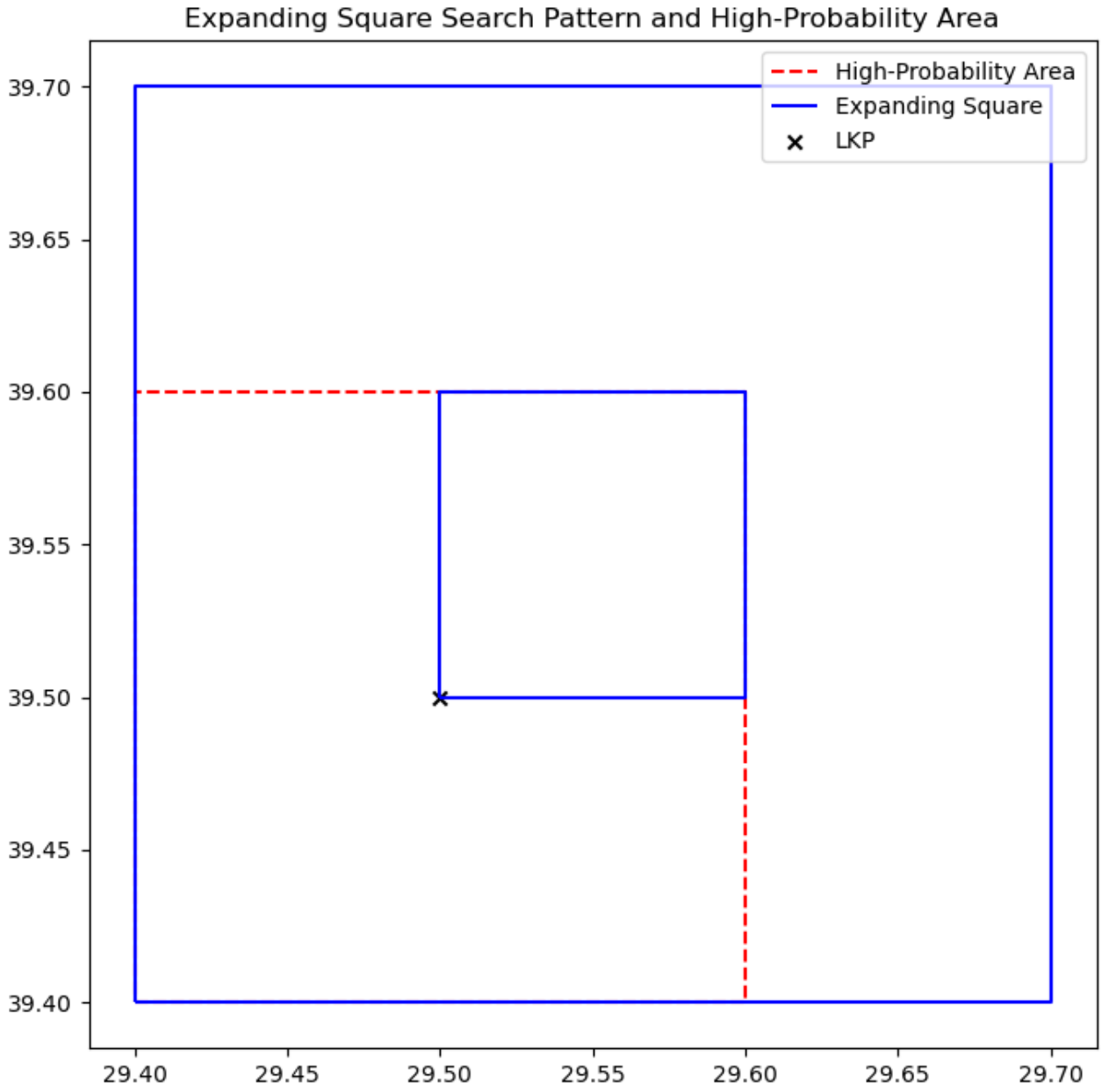
```
high_prob_area = gpd.GeoDataFrame(geometry=[Polygon(high_prob_area_coords)],

# Step 2: Generate Search Patterns
st.sidebar.subheader("Search Patterns")
pattern_type = st.sidebar.selectbox("Select Search Pattern", ["Expanding Squ

lkp = Point(29.5, 39.5)
search_area = Polygon([(29.3, 39.3), (29.7, 39.3), (29.7, 39.7), (29.3, 39.7

if pattern_type == "Expanding Square":
    expanding_square_coords = [
        [(29.5, 39.5), (29.6, 39.5), (29.6, 39.6), (29.5, 39.6), (29.5, 39.5
        [(29.4, 39.4), (29.7, 39.4), (29.7, 39.7), (29.4, 39.7), (29.4, 39.4
    ]
    search_pattern = gpd.GeoDataFrame(geometry=[Polygon(coords) for coords i

elif pattern_type == "Parallel Track":
    parallel_tracks = [
        LineString([(29.3, y), (29.7, y)]) for y in [39.3, 39.4, 39.5, 39.6,
    ]
    search_pattern = gpd.GeoDataFrame(geometry=parallel_tracks, crs="EPSG:43

# Step 3: Visualize Search Patterns and High-Probability Area
st.subheader("Search Pattern Visualization")
fig, ax = plt.subplots(figsize=(10, 8))

high_prob_area.boundary.plot(ax=ax, color="red", linestyle="--", label="High
search_pattern.boundary.plot(ax=ax, color="blue", label=pattern_type)
gpd.GeoSeries([lkp]).plot(ax=ax, color="black", label="LKP", marker="x")

ax.set_title(f"{pattern_type} Search Pattern and High-Probability Area")
ax.legend()
st.pyplot(fig)

# Step 4: Analyze Coverage
st.subheader("Coverage Analysis")
coverage = search_pattern.intersection(high_prob_area.unary_union).area.sum(
st.write(f"Coverage of the {pattern_type}: **{coverage:.2f} area units**")

# Step 5: Add Probability Map Visualization (Optional)
st.sidebar.subheader("Probability Map")
show_prob_map = st.sidebar.checkbox("Show Probability Map", value=False)

if show_prob_map:
    with rasterio.open("mcda_probability_map.tif") as src:
        prob_map = src.read(1)
        plt.figure(figsize=(10, 8))
        plt.imshow(prob_map, cmap="viridis", origin="upper")
        plt.colorbar(label="Probability")
        plt.title("Probability Map")
        plt.xlabel("Longitude")
        plt.ylabel("Latitude")
        st.pyplot()
```

```
2025-01-19 04:02:50.484 WARNING streamlit.runtime.scriptrunner_utils.script_
run_context: Thread 'MainThread': missing ScriptRunContext! This warning can
be ignored when running in bare mode.
```

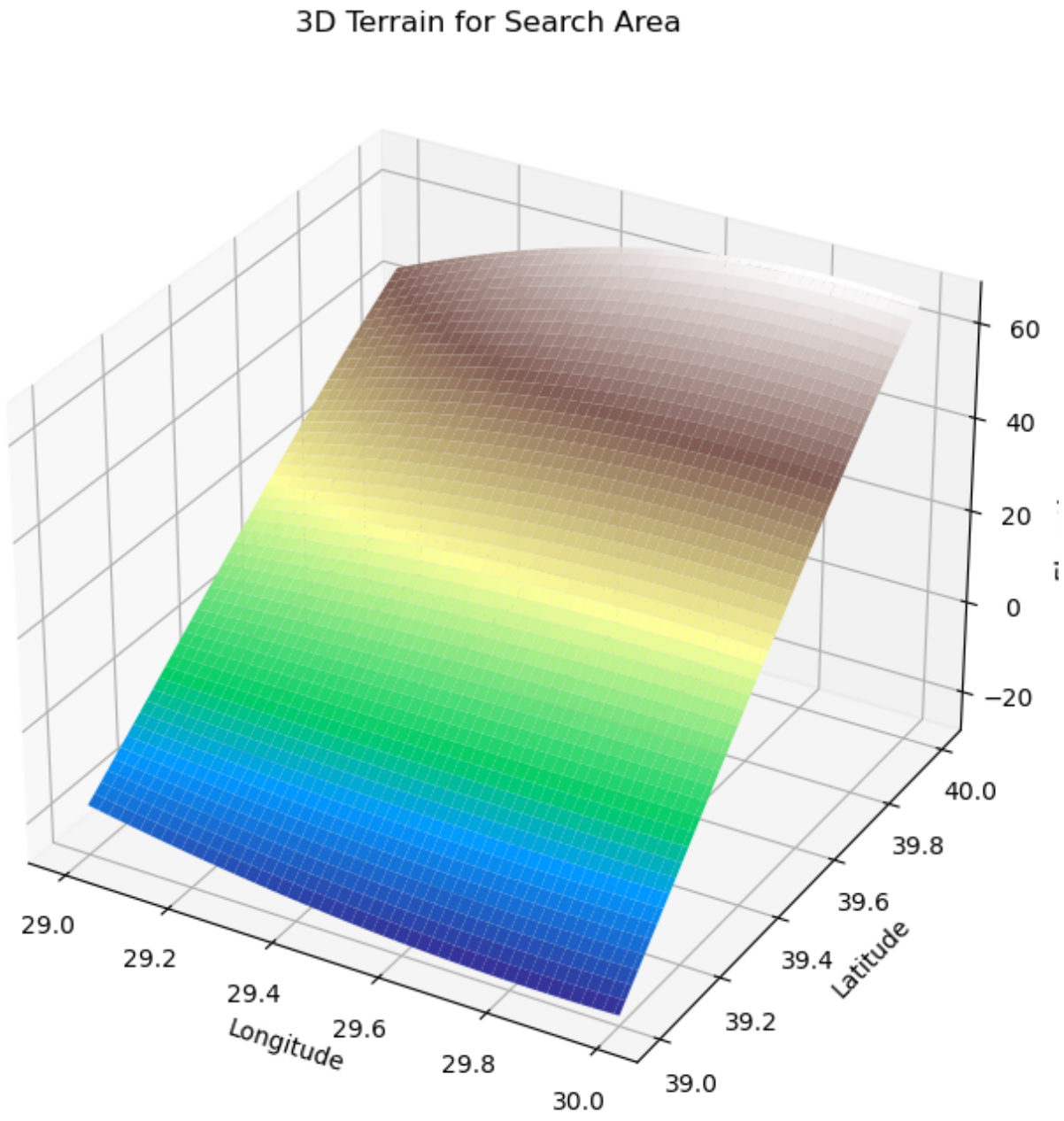
```
2025-01-19 04:02:50.534
  Warning: to view this Streamlit app on a browser, run it with the followin
g
  command:

    streamlit run /Users/lasyatummala/anaconda3/lib/python3.11/site-package
s/ipykernel_launcher.py [ARGUMENTS]
2025-01-19 04:02:50.535 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.535 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.536 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.537 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.537 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.538 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.538 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.538 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.539 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.540 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.541 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.542 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.542 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.542 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.543 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.543 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.544 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.544 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.544 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.545 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.545 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.545 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.545 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.546 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.546 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
```

```
2025-01-19 04:02:50.575 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.576 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.577 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.577 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.578 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.579 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.580 Session state does not function when running a scrip
t without `streamlit run`
2025-01-19 04:02:50.581 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.581 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.584 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.585 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.765 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.919 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.919 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.920 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.920 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
/var/folders/fp/drm5pw1d13x1n4wvd2q14v0m0000gn/T/ipykernel_26763/539321407.p
y:49: DeprecationWarning: The 'unary_union' attribute is deprecated, use the
'union_all()' method instead.
  coverage = search_pattern.intersection(high_prob_area.unary_union).area.su
m()
/var/folders/fp/drm5pw1d13x1n4wvd2q14v0m0000gn/T/ipykernel_26763/539321407.p
y:49: UserWarning: Geometry is in a geographic CRS. Results from 'area' are
likely incorrect. Use 'GeoSeries.to_crs()' to re-project geometries to a pro
jected CRS before this operation.

  coverage = search_pattern.intersection(high_prob_area.unary_union).area.su
m()
2025-01-19 04:02:50.941 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.942 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.942 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.943 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.943 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.943 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.944 Thread 'MainThread': missing ScriptRunContext! This
```

```
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.944 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.945 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.946 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
2025-01-19 04:02:50.946 Thread 'MainThread': missing ScriptRunContext! This
warning can be ignored when running in bare mode.
```



In [88]:
```
# Simulated elevation data for a search grid
x = np.linspace(29.0, 30.0, 100)  # Longitude
y = np.linspace(39.0, 40.0, 100)  # Latitude
xv, yv = np.meshgrid(x, y)
elevation = np.sin(xv) * np.cos(yv) * 100  # Simulated terrain elevation

# Plot the 3D terrain
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection="3d")
ax.plot_surface(xv, yv, elevation, cmap="terrain", edgecolor="none")
ax.set_title("3D Terrain for Search Area")
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")
ax.set_zlabel("Elevation (m)")
plt.show()
```



In [ ]: