Project Report for Project 1: JSON to Shapefile Conversion

---

1. Introduction

Project Overview:

This project is focused on converting a JSON file containing land value data for New Orleans into a shapefile format, suitable for use in GIS applications, particularly ArcGIS Pro. The data in the JSON file is in the form of Well-Known Text (WKT) for geometries and includes several attribute fields with information about land values. However, ArcGIS Pro cannot directly import WKT geometries from a JSON file. The goal of this project is to write a Python script that automates the conversion of this data into a shapefile, which can be subsequently visualized and analyzed in ArcGIS Pro.

Objective:

The main objective is to create a Python script that:

1. Reads data from a JSON file containing WKT geometries and related attribute fields.
2. Converts the data into a shapefile format.
3. Allows the user to visualize the shapefile data in ArcGIS Pro, modify its symbology, and print a layout to PDF for reporting purposes.

---

2. Data Format and Description

The data for this project is provided in a JSON file named `no_tax.json`, which is structured into two main sections:

- **Meta Section:**
  - The "meta" section provides metadata about the dataset, including descriptions of the fields. It outlines what each field represents, such as land value, location, or zoning information. These descriptions are crucial for understanding the structure and meaning of the data.
- **Data Section:**
  - The "data" section contains the actual records, with each record representing a geographic feature. Each record includes:
    - **WKT Geometry:** The geometric representation of the feature (polygon) in Well-Known Text (WKT) format.
    - **Attribute Fields:** Associated attribute values such as land value, zoning type, and other metadata.

The task is to extract the geometries and attributes, convert them to a format usable by ArcGIS, and then store them in a shapefile.

---

3. Code Explanation and Walkthrough

This section explains the Python code that was developed to import the JSON data into ArcGIS and convert it into a shapefile.

1. **Importing Necessary Libraries:**

```
python
Copy code
import json
import arcpy
import os
```

The code begins by importing essential libraries:

- `json` is used to load and parse the JSON data.
- `arcpy` is the ArcGIS Python library, used to manipulate geospatial data and perform spatial analysis.
- `os` is used for handling file paths and checking if the shapefile already exists.

2. **Reading and Processing the JSON File:** The script defines the `importNoTaxJSON()` function, which reads the JSON file and processes the WKT geometries:

```
python
Copy code
with open(json_file,'r') as file:
    tax_json = json.load(file)
```

The JSON file is opened and loaded into the `tax_json` variable. The code then processes each geometry (WKT) and converts it using ArcPy's `FromWKT` function:

```
python
Copy code
for row in tax_json['data']:
    row[8] = arcpy.FromWKT(row[8])
```

3. **Creating a Shapefile:** The shapefile is created using `arcpy.management.CreateFeatureclass()`. The script checks if the shapefile already exists and deletes it if necessary:

```
python
Copy code
fcname = out_fc
fc_fullname = os.path.join(workspace, fcname)
if arcpy.Exists(fc_fullname):
    arcpy.management.Delete(fc_fullname)
```

A new shapefile is created with the appropriate geometry type (polygon) and spatial reference system (EPSG: 4236, WGS 84).

4. **Adding Fields to the Shapefile:** The script iterates over the field names defined in the "meta" section of the JSON file and adds these fields to the shapefile:

```
python
Copy code
field_type                                                                =
['TEXT','TEXT','LONG','LONG','TEXT','LONG','TEXT','TEXT','TEXT','TEXT','TEXT',
'TEXT','TEXT']
field_names = []
```

It ensures that the field names are formatted correctly by removing spaces and replacing periods with underscores.

5. **Inserting Data into the Shapefile:** The script uses an `InsertCursor` to insert each row from the JSON data into the shapefile:

```
python
Copy code
with arcpy.da.InsertCursor(fc_fullname, field_names) as cursor:
    for row in tax_json['data']:
        new_row = []
        for ind, value in enumerate(row):
            if ind == 8: continue
            if value == None: value = ""
            new_row.append(value)
        new_row.append(row[8])  # Add geometry (WKT)
        cursor.insertRow(new_row)
```

The `InsertCursor` adds each feature's attributes, and the geometry is added in the last position of the row.

---

4. Interface Design

A Python Toolbox (pyt) was created to provide a graphical interface for the user to interact with the script within ArcGIS Pro. The toolbox allows users to:

1. Select the input JSON file via a file dialog.
2. Specify the output shapefile name and location.

Interface Design Details:

- The user interface is simple and intuitive, allowing for easy integration of the script within ArcGIS Pro without needing to interact with the code directly.

- ArcPy tools were used to create the Python Toolbox, making it compatible with the ArcGIS Pro environment.

---

5. Data Visualization and Layout

Once the shapefile is created, it can be visualized and further analyzed in ArcGIS Pro. The following steps outline the process of creating a map and adding the shapefile for visualization:

1. **Create a New Map:**
   - A new map is created within ArcGIS Pro, providing a blank canvas for the shapefile visualization.
2. **Add Shapefile to Map:**
   - The newly generated shapefile is added to the map for visualization.
3. **Change Symbology:**
   - The symbology of the shapefile is modified to display the land value data effectively. Different categories, such as zoning types, can be used to classify the land based on color.
4. **Add a Layout:**
   - A layout is created to display the map. The layout includes essential map elements such as a map frame, title, legend, and north arrow.
   - Additional elements like scale bars, text annotations, and a legend are added to improve the clarity of the map.
5. **Print Layout to PDF:**
   - The completed layout is exported to a PDF for easy sharing and printing.

---

6. Final Layout and Screenshots

Final Layout (Project1.pdf): The final map layout should display the shapefile with proper symbology, and the layout should include all necessary elements. Example layout elements include:

- **Map Title:** A descriptive title indicating the project’s focus .
- **Legend:** A legend explaining the symbology used in the map.
- **Scale Bar:** To show the scale of the map.
- **North Arrow:** To show the orientation of the map.

---

7. Conclusion

Summary of Work: The project successfully converts JSON data, containing land value information and geometries, into a shapefile format that can be used within ArcGIS Pro. The Python script automates the process of reading the JSON file, creating a shapefile, and inserting both geometries and attributes.

Instructions for Use:

1. Install ArcGIS Pro with the Python and ArcPy libraries.
2. Place the `no_tax.json` file in the correct directory.
3. Run the Python script through the Python Toolbox in ArcGIS Pro. Select the input JSON file and specify the output shapefile location.
4. Open the resulting shapefile in ArcGIS Pro for visualization and analysis.

---

8. GitHub Repository

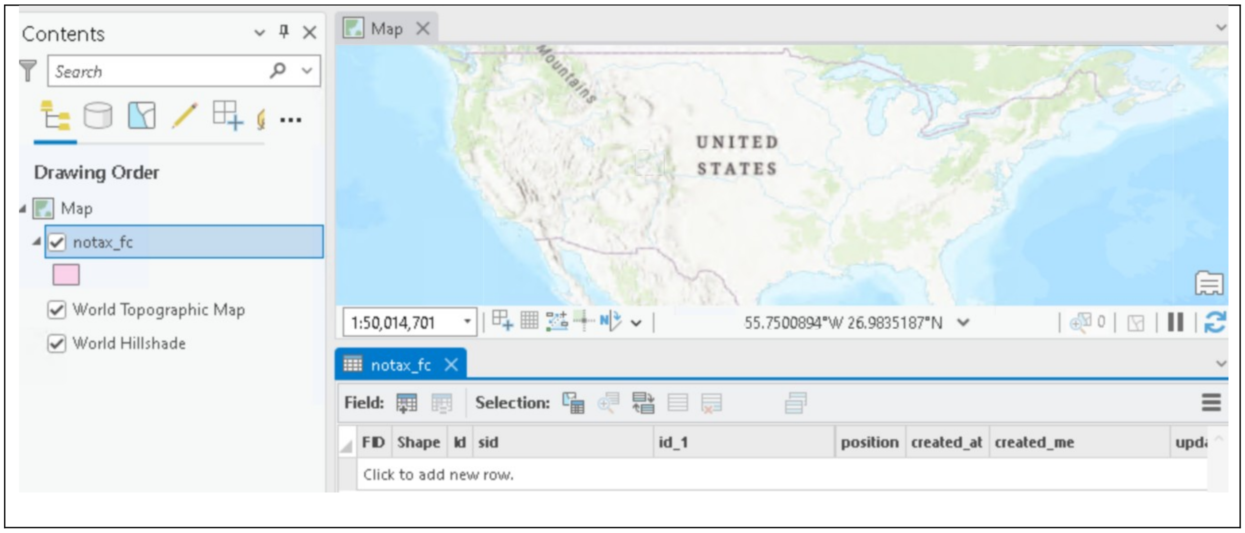
The following link is the GitHub repository, where the project files, code, and documentation are hosted:

- https://github.com/shabnamm7/geog4057_shabnam/tree/main/FinalProject/Project1

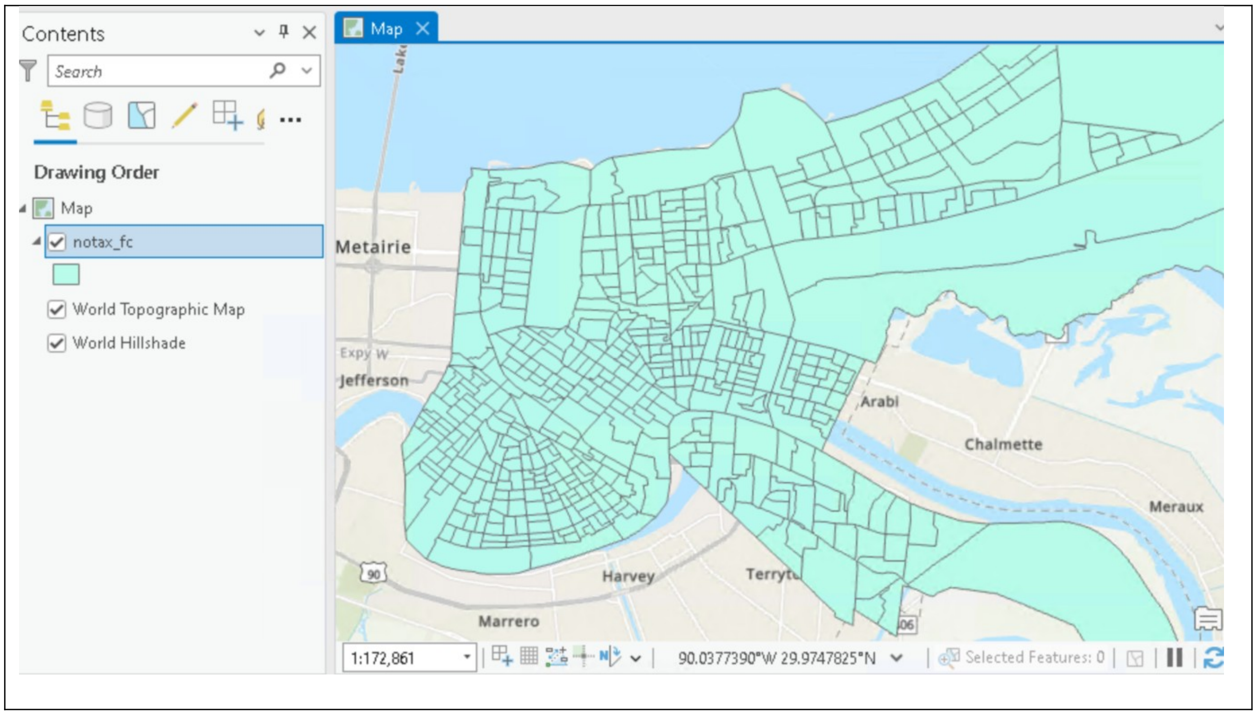
Appendices

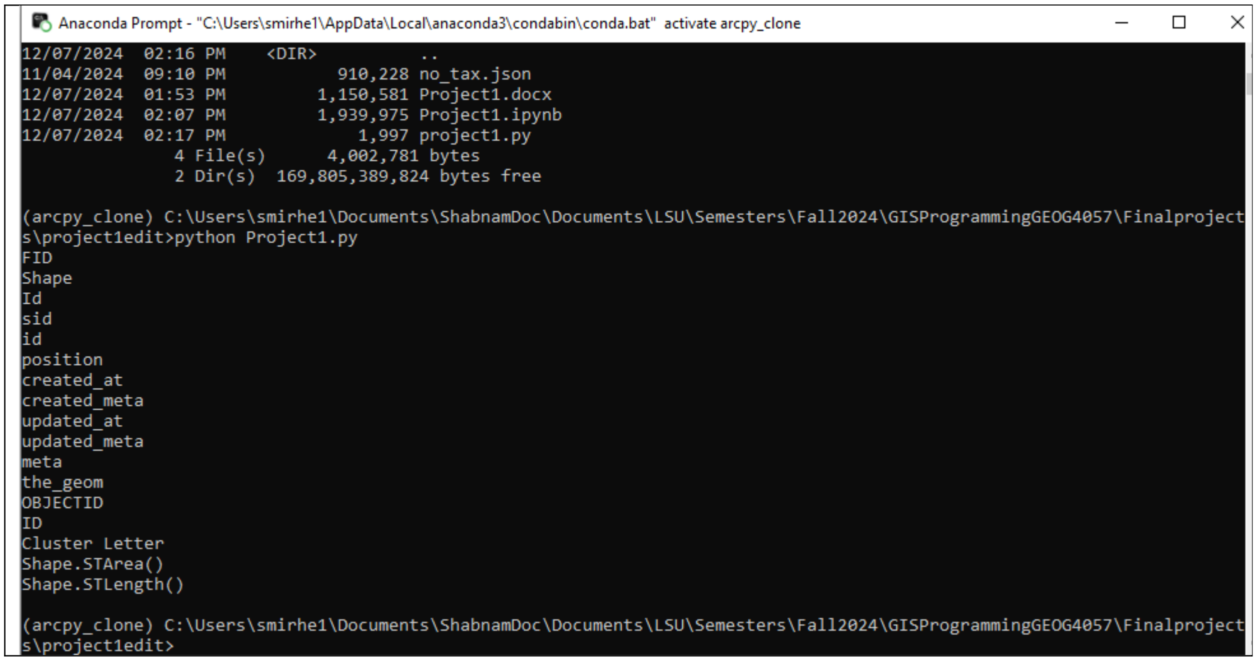
1. **Code:** Include the full Python code in the appendices or as a separate document.
2. **References:** List any references or resources you consulted during the development of the project, such as the ArcGIS documentation or tutorials on JSON handling in Python.

Screenshots of the project:



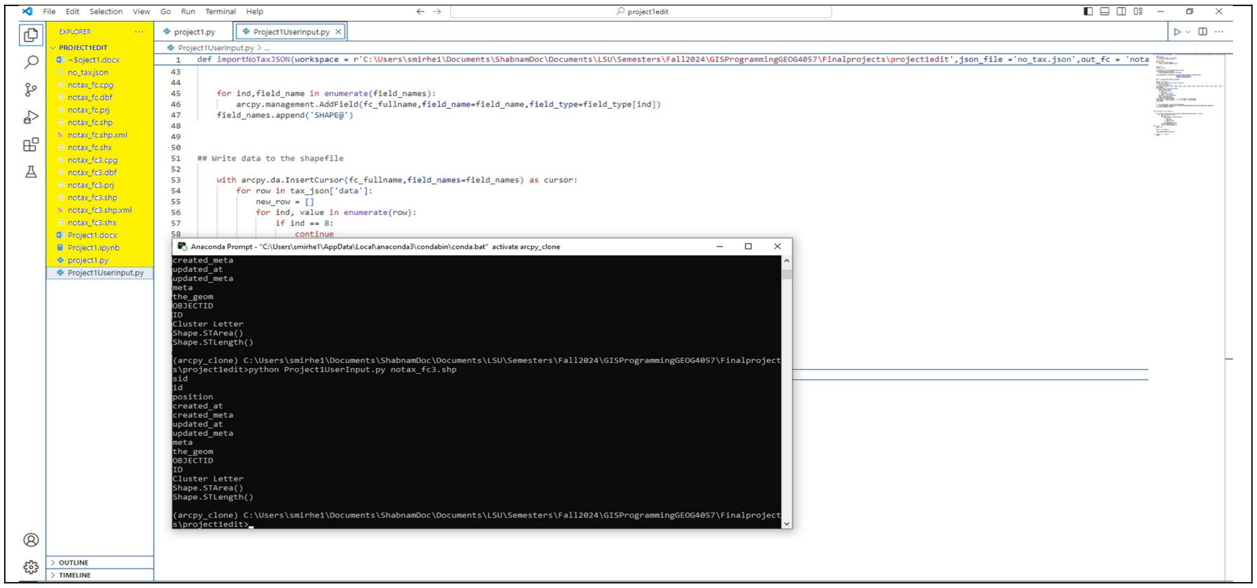
FID	Shape	Id	sid	id_1	position	created_at	created_me	upda
Click to add new row.								





Anaconda Prompt - "C:\Users\smirhe1\AppData\Local\anaconda3\condabin\conda.bat" activate arcpy_clone

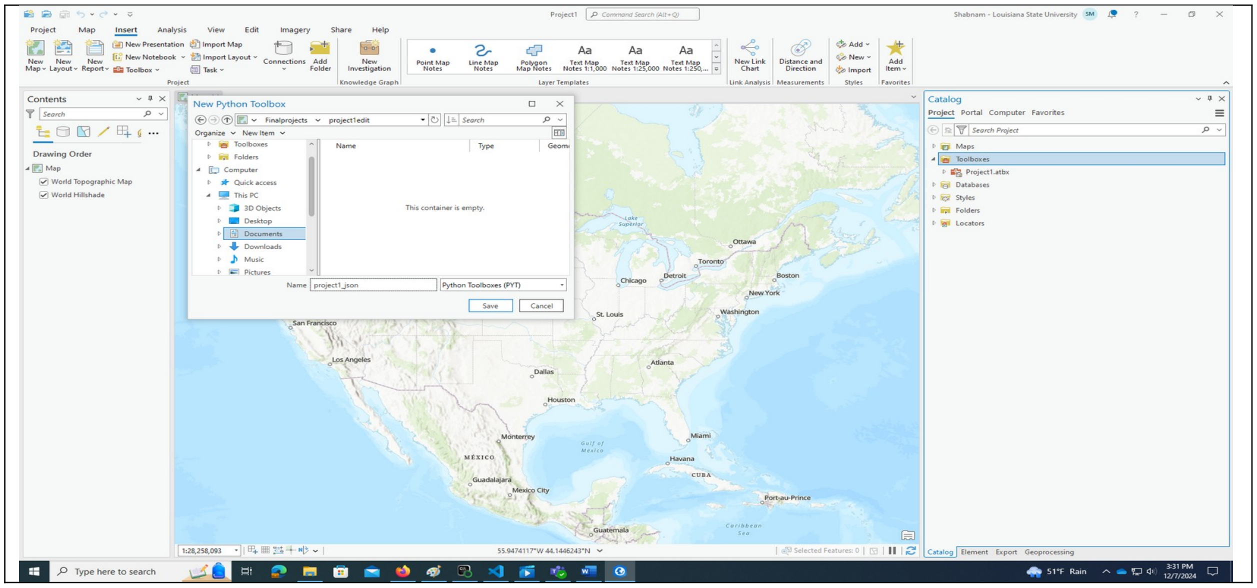
```
12/07/2024  02:16 PM    <DIR>          ..
11/04/2024  09:10 PM           910,228 no_tax.json
12/07/2024  01:53 PM         1,150,581 Project1.docx
12/07/2024  02:07 PM         1,939,975 Project1.ipynb
12/07/2024  02:17 PM             1,997 project1.py
               4 File(s)      4,002,781 bytes
               2 Dir(s)  169,805,389,824 bytes free

(arcpy_clone) C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall2024\GISProgrammingGEOG4057\Finalproject
s\project1edit>python Project1.py
FID
Shape
Id
sid
id
position
created_at
created_meta
updated_at
updated_meta
meta
the_geom
OBJECTID
ID
Cluster Letter
Shape.STArea()
Shape.STLength()

(arcpy_clone) C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall2024\GISProgrammingGEOG4057\Finalproject
s\project1edit>
```



```python
def importNoTaxJSON(workspace = r'C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall2024\GISProgrammingGEOG4057\Finalprojects\project1edit',json_file ='no_tax.json',out_fc = 'nota


    for ind,field_name in enumerate(field_names):
        arcpy.management.AddField(fc_fullname,field_name=field_name,field_type=field_type[ind])
    field_names.append('SHAPE@')



## Write data to the shapefile

    with arcpy.da.InsertCursor(fc_fullname,field_names=field_names) as cursor:
        for row in tax_json['data']:
            new_row = []
            for ind, value in enumerate(row):
                if ind == 8:
                    continue
```

Anaconda Prompt - "C:\Users\smirhe1\AppData\Local\anaconda3\condabin\conda.bat" activate arcpy_clone

```
created_meta
updated_at
updated_meta
meta
the_geom
OBJECTID
ID
Cluster Letter
Shape.STArea()
Shape.STLength()

(arcpy_clone) C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall2024\GISProgrammingGEOG4057\Finalproject
s\project1edit>python Project1UserInput.py notax_fc3.shp
sid
id
position
created_at
created_meta
updated_at
updated_meta
meta
the_geom
OBJECTID
ID
Cluster Letter
Shape.STArea()
Shape.STLength()

(arcpy_clone) C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall2024\GISProgrammingGEOG4057\Finalproject
s\project1edit>
```


New Python Toolbox

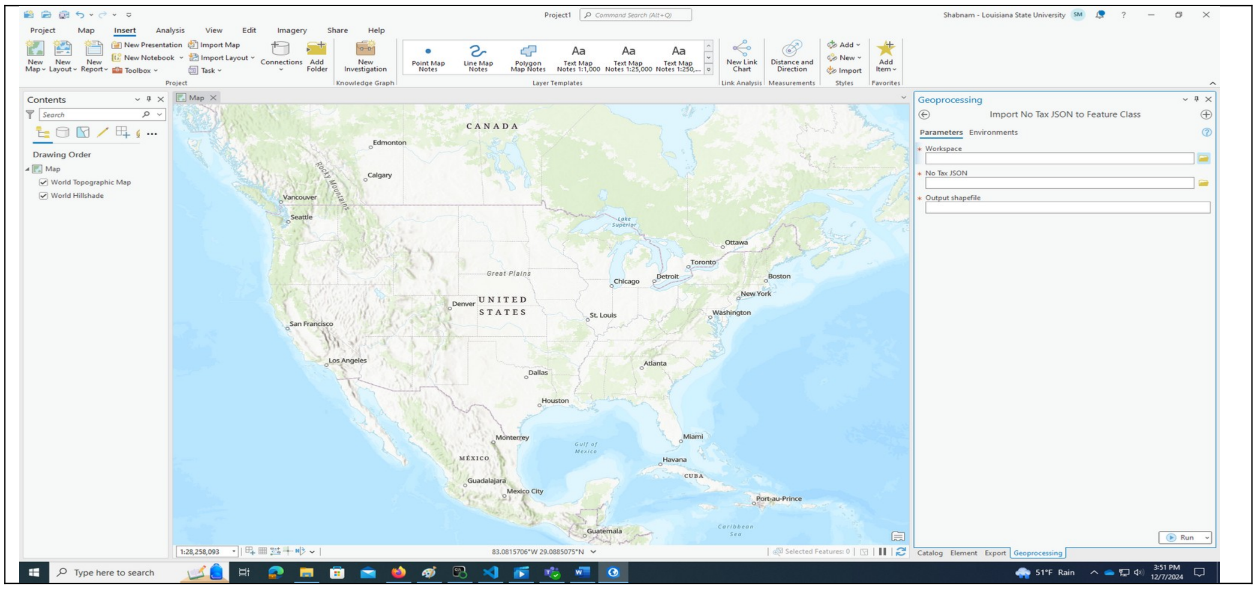
Finalprojects > project1edit

This container is empty.

Name: project1_json

Python Toolboxes (PYT)

Save Cancel


Geoprocessing

Import No Tax JSON to Feature Class

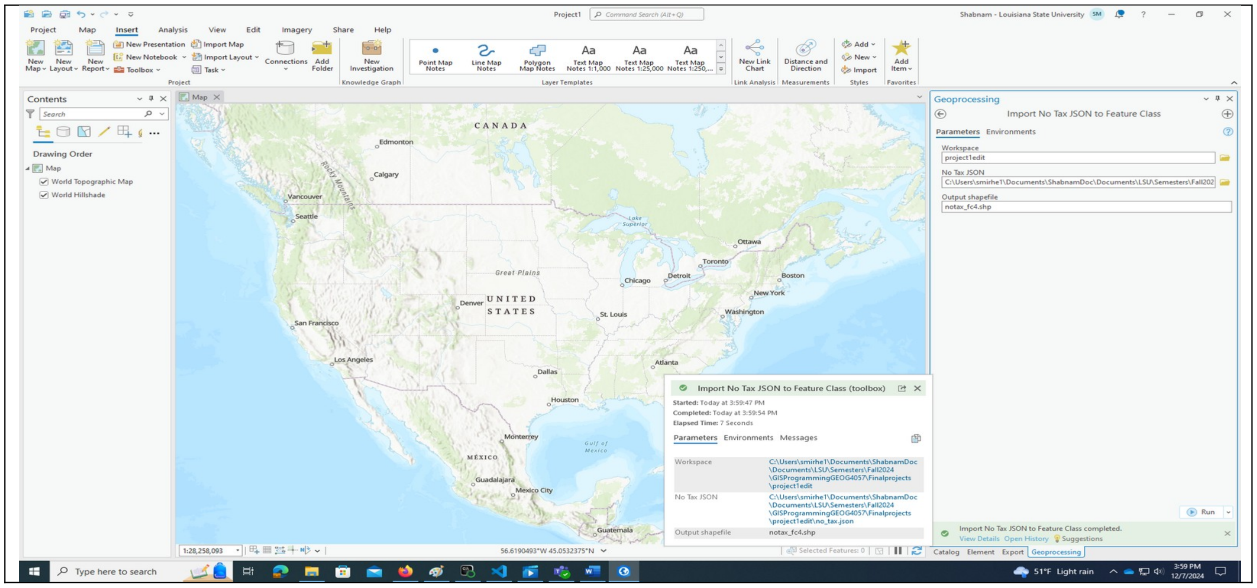
Parameters Environments

Workspace

No Tax JSON

Output shapefile

Run



Geoprocessing

Import No Tax JSON to Feature Class

Parameters Environments

Workspace: project1edit

No Tax JSON: C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall202

Output shapefile: notax_fc4.shp

Import No Tax JSON to Feature Class completed.

Import No Tax JSON to Feature Class (toolbox)

Started: Today at 3:59:47 PM
Completed: Today at 3:59:54 PM
Elapsed Time: 7 Seconds

Parameters Environments Messages

Workspace	C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall2024\GISProgrammingGEOG4057\Finalprojects\project1edit
No Tax JSON	C:\Users\smirhe1\Documents\ShabnamDoc\Documents\LSU\Semesters\Fall2024\GISProgrammingGEOG4057\Finalprojects\project1edit\no_tax.json
Output shapefile	notax_fc4.shp