Collected Code Files

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\app.py

from flask import Flask, render\_template, request, url\_for, send\_file, redirect, jsonify  
from sqlalchemy import create\_engine, MetaData, Table, and\_, select, distinct, func, text, or\_  
import os  
import glob2  
from datetime import datetime  
import shutil  
  
PROJECT\_ROOT = os.path.dirname(os.path.abspath(\_\_file\_\_))  
  
def row\_to\_dict(row):  
 """  
 Convert SQLAlchemy Row object to dictionary, handling different SQLAlchemy versions  
 """  
 try:  
 if hasattr(row, '\_mapping'):  
 return dict(row.\_mapping)  
 else:  
 return dict(row)  
 except (ValueError, TypeError):  
 # Fallback for SQLAlchemy Row objects  
 return {key: row[key] for key in row.keys()}  
  
app = Flask(\_\_name\_\_)  
DATABASE = 'elements.db'  
UPLOAD\_FOLDER = 'sampleDataset'  
app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER  
  
  
# Custom filter for datetime formatting  
@app.template\_filter('datetime')  
def datetime\_filter(timestamp):  
 return datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')  
  
DATABASE\_URL = 'sqlite:///elements.db'  
engine = create\_engine(DATABASE\_URL)  
metadata = MetaData()  
  
# Reflect only the tables that exist  
projects\_table = Table('projects', metadata, autoload\_with=engine)  
areas\_table = Table('areas', metadata, autoload\_with=engine)  
  
def parse\_point(s):  
 """  
 Parse coordinate string with support for various separators and formats.  
 Supports: '/', ',', ':', ';', '|', ' ', '\t', '\\', and combinations  
 Also handles WGS84 format and other coordinate system prefixes  
 Handles complex formats like:  
 - WGS84 UTM 36N 735712 E / 3563829 N  
 - WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N  
   
 Returns: (x, y) if successful, or (None, error\_message) if failed  
 """  
 try:  
 s = str(s).strip()  
   
 # Check for empty or whitespace-only input  
 if not s:  
 return None, "Empty coordinate string provided"  
   
 # Handle complex WGS84 UTM format: "WGS84 UTM 36N 735712 E / 3563829 N"  
 if 'WGS84 UTM' in s.upper():  
 import re  
 # Pattern: WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]  
 utm\_pattern = r'WGS84\s+UTM\s+(\d+[NS])\s+(\d+)\s\*[EW]\s\*/\s\*(\d+)\s\*[NS]'  
 match = re.search(utm\_pattern, s, re.IGNORECASE)  
 if match:  
 try:  
 zone = match.group(1)  
 easting = float(match.group(2))  
 northing = float(match.group(3))  
 return (easting, northing), None  
 except ValueError as e:  
 return None, f"Invalid UTM coordinates in '{s}': {str(e)}"  
 else:  
 return None, f"Invalid WGS84 UTM format. Expected: 'WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]'"  
   
 # Handle complex WGS84 Geographic format: "WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N"  
 if 'WGS84 GEO' in s.upper():  
 import re  
 # Pattern: WGS84 Geo [deg]° [min]' [sec]" [E/W] / [deg]° [min]' [sec]" [N/S]  
 geo\_pattern = r'WGS84\s+GEO\s+(\d+)°\s\*(\d+)\'\s\*([\d.]+)"\s\*[EW]\s\*/\s\*(\d+)°\s\*(\d+)\'\s\*([\d.]+)"\s\*[NS]'  
 match = re.search(geo\_pattern, s, re.IGNORECASE)  
 if match:  
 try:  
 # Convert DMS to decimal degrees  
 lon\_deg, lon\_min, lon\_sec = float(match.group(1)), float(match.group(2)), float(match.group(3))  
 lat\_deg, lat\_min, lat\_sec = float(match.group(4)), float(match.group(5)), float(match.group(6))  
   
 # Check if longitude is East or West  
 if 'W' in s.upper():  
 lon\_deg = -lon\_deg  
 if 'S' in s.upper():  
 lat\_deg = -lat\_deg  
   
 # Convert to decimal degrees  
 lon\_decimal = lon\_deg + (lon\_min / 60) + (lon\_sec / 3600)  
 lat\_decimal = lat\_deg + (lat\_min / 60) + (lat\_sec / 3600)  
   
 return (lon\_decimal, lat\_decimal), None  
 except ValueError as e:  
 return None, f"Invalid geographic coordinates in '{s}': {str(e)}"  
 else:  
 return None, f"Invalid WGS84 Geographic format. Expected: 'WGS84 Geo [deg]° [min]' [sec]\" [E/W] / [deg]° [min]' [sec]\" [N/S]'"  
   
 # Handle simple WGS84 and other coordinate system prefixes  
 if s.upper().startswith(('WGS', 'EPSG', 'UTM', 'GEO', 'PROJ')):  
 # Extract coordinates after the prefix  
 # Look for common patterns like "WGS84: 123.456, 789.012" or "UTM 36N: 123456, 789012"  
 import re  
 # Match coordinates after any prefix  
 coord\_match = re.search(r'[:\s]+([-\d.,\s]+)$', s)  
 if coord\_match:  
 s = coord\_match.group(1).strip()  
 else:  
 return None, f"Invalid coordinate system format. Expected: '[SYSTEM]: [x], [y]' or '[SYSTEM] [x], [y]'"  
   
 # Remove any parentheses, brackets, or quotes  
 s = s.strip('()[]{}"\'\'')  
   
 # Try multiple separators in order of preference  
 separators = ['/', ',', ':', ';', '|', '\\', '\t']  
   
 # First try exact separators  
 for sep in separators:  
 if sep in s:  
 parts = s.split(sep, 1) # Split only on first occurrence  
 if len(parts) == 2:  
 x\_str, y\_str = parts[0].strip(), parts[1].strip()  
 # Try to convert to float  
 try:  
 return (float(x\_str), float(y\_str)), None  
 except ValueError:  
 continue  
   
 # If no separator found, try splitting on whitespace  
 if ' ' in s:  
 parts = s.split()  
 if len(parts) >= 2:  
 try:  
 return (float(parts[0]), float(parts[1])), None  
 except ValueError:  
 pass  
   
 # Try regex pattern for coordinates with optional spaces and various separators  
 import re  
 # Pattern: number, optional spaces, separator, optional spaces, number  
 coord\_pattern = r'([-+]?\d\*\.?\d+)\s\*[\/,:;|\t\\]\s\*([-+]?\d\*\.?\d+)'  
 match = re.search(coord\_pattern, s)  
 if match:  
 try:  
 return (float(match.group(1)), float(match.group(2))), None  
 except ValueError:  
 pass  
   
 # Try pattern for coordinates separated by whitespace  
 space\_pattern = r'([-+]?\d\*\.?\d+)\s+([-+]?\d\*\.?\d+)'  
 match = re.search(space\_pattern, s)  
 if match:  
 try:  
 return (float(match.group(1)), float(match.group(2))), None  
 except ValueError:  
 pass  
   
 # If we get here, no valid format was found  
 return None, f"Invalid coordinate format: '{s}'. Expected formats: 'x,y', 'x/y', 'x:y', 'WGS84 UTM 36N 735712 E / 3563829 N', 'WGS84 Geo 35° 30' 0.11\" E / 32° 11' 9.88\" N', etc."  
 except Exception as e:  
 return None, f"Error parsing coordinates '{s}': {str(e)}"  
  
def calculate\_area\_size(xmin, ymin, xmax, ymax):  
 """Calculate the area size in square meters using UTM coordinates"""  
 width = abs(xmax - xmin)  
 height = abs(ymax - ymin)  
 return width \* height  
  
def calculate\_overlap\_percentage(area\_xmin, area\_ymin, area\_xmax, area\_ymax, query\_xmin, query\_ymin, query\_xmax, query\_ymax):  
 """Calculate the percentage of area that overlaps with the query rectangle"""  
 # Calculate intersection  
 intersect\_xmin = max(area\_xmin, query\_xmin)  
 intersect\_ymin = max(area\_ymin, query\_ymin)  
 intersect\_xmax = min(area\_xmax, query\_xmax)  
 intersect\_ymax = min(area\_ymax, query\_ymax)  
  
 # Check if there's an intersection  
 if intersect\_xmin >= intersect\_xmax or intersect\_ymin >= intersect\_ymax:  
 return 0.0  
  
 # Calculate areas  
 area\_size = (area\_xmax - area\_xmin) \* (area\_ymax - area\_ymin)  
 intersect\_size = (intersect\_xmax - intersect\_xmin) \* (intersect\_ymax - intersect\_ymin)  
  
 if area\_size == 0:  
 return 0.0  
  
 return (intersect\_size / area\_size) \* 100.0  
  
@app.route('/api/add\_project', methods=['POST'])  
def api\_add\_project():  
 data = request.get\_json()  
   
 if not data:  
 return jsonify({"error": "No JSON data provided"}), 400  
   
 required\_fields = ['uuid', 'project\_name', 'user\_name', 'date', 'file\_location', 'paper\_size', 'description']  
 missing\_fields = [f for f in required\_fields if f not in data]  
   
 if missing\_fields:  
 return jsonify({"error": f"Missing fields: {', '.join(missing\_fields)}"}), 400  
   
 try:  
 with engine.begin() as conn:  
 # Insert project  
 conn.execute(projects\_table.insert().values(  
 uuid=data['uuid'],  
 project\_name=data['project\_name'],  
 user\_name=data['user\_name'],  
 date=data['date'],  
 file\_location=data['file\_location'],  
 paper\_size=data['paper\_size'],  
 description=data['description']  
 ))  
   
 # Insert areas if provided  
 if 'areas' in data and isinstance(data['areas'], list):  
 for area\_data in data['areas']:  
 area\_required\_fields = ['xmin', 'ymin', 'xmax', 'ymax', 'scale']  
 area\_missing\_fields = [f for f in area\_required\_fields if f not in area\_data]  
   
 if area\_missing\_fields:  
 return jsonify({"error": f"Missing area fields: {', '.join(area\_missing\_fields)}"}), 400  
   
 conn.execute(areas\_table.insert().values(  
 project\_id=data['uuid'],  
 xmin=area\_data['xmin'],  
 ymin=area\_data['ymin'],  
 xmax=area\_data['xmax'],  
 ymax=area\_data['ymax'],  
 scale=area\_data['scale']  
 ))  
   
 return jsonify({"message": "Project added successfully", "uuid": data['uuid']}), 201  
 except Exception as e:  
 return jsonify({"error": str(e)}), 500  
  
@app.route('/api/get\_project/<uuid>', methods=['GET'])  
def api\_get\_project(uuid):  
 try:  
 with engine.connect() as conn:  
 # Get project details  
 project\_result = conn.execute(  
 select(projects\_table).where(projects\_table.c.uuid == uuid)  
 ).first()  
   
 if not project\_result:  
 return jsonify({"error": "Project not found"}), 404  
   
 project\_dict = row\_to\_dict(project\_result)  
   
 # Get associated areas  
 areas\_result = conn.execute(  
 select(areas\_table).where(areas\_table.c.project\_id == uuid)  
 ).fetchall()  
   
 areas\_list = [row\_to\_dict(area) for area in areas\_result]  
 project\_dict['areas'] = areas\_list  
   
 return jsonify(project\_dict), 200  
 except Exception as e:  
 return jsonify({"error": str(e)}), 500  
  
@app.route('/', methods=['GET', 'POST'])  
def index():  
 results = None  
 error = None  
 # Query unique user names for the dropdown  
 with engine.connect() as conn:  
 user\_names = [row[0] for row in conn.execute(select(projects\_table.c.user\_name).distinct())]  
 selected\_user\_names = []  
  
 if request.method == 'POST':  
 # This block handles the main search form submission  
 filters = []  
 join\_areas = False  
 # Parse spatial box  
 bottom\_left = request.form.get('bottom\_left', '').strip()  
 top\_right = request.form.get('top\_right', '').strip()  
 # Removed: relative\_size\_enabled, size\_percentage, inside\_enabled, outside\_enabled, percentage\_overlap\_enabled, overlap\_percentage  
  
 if bottom\_left and top\_right:  
 bl\_result = parse\_point(bottom\_left)  
 tr\_result = parse\_point(top\_right)  
   
 # Check for parsing errors  
 if bl\_result[1] is not None: # Error in bottom\_left  
 error = f'Bottom Left: {bl\_result[1]}'  
 elif tr\_result[1] is not None: # Error in top\_right  
 error = f'Top Right: {tr\_result[1]}'  
 elif not bl\_result[0] or not tr\_result[0]: # No coordinates returned  
 error = 'Invalid input format. Please use X/Y or X,Y for both points.'  
 else:  
 xmin, ymin = bl\_result[0]  
 xmax, ymax = tr\_result[0]  
 if xmin >= xmax or ymin >= ymax:  
 error = 'Bottom Left must be southwest (smaller X and Y) of Top Right. Please check your input.'  
 else:  
 join\_areas = True  
 # Only use the default INSIDE spatial filter  
 inside\_filters = [  
 areas\_table.c.xmin >= xmin,  
 areas\_table.c.xmax <= xmax,  
 areas\_table.c.ymin >= ymin,  
 areas\_table.c.ymax <= ymax  
 ]  
 filters.append(and\_(\*inside\_filters))  
 # Parse other filters  
 uuid = request.form.get('uuid', '').strip()  
 if uuid:  
 filters.append(projects\_table.c.uuid.ilike(f"{uuid}%"))  
 user\_name\_list = request.form.getlist('user\_name')  
 selected\_user\_names = [n for n in user\_name\_list if n]  
 if selected\_user\_names:  
 filters.append(or\_(\*[projects\_table.c.user\_name.ilike(f"{n}%") for n in selected\_user\_names]))  
 paper\_size = request.form.get('paper\_size', '').strip()  
 custom\_height = request.form.get('custom\_height', '').strip()  
 custom\_width = request.form.get('custom\_width', '').strip()  
  
 if paper\_size:  
 if paper\_size == 'custom' and custom\_height and custom\_width:  
 try:  
 height\_cm = float(custom\_height)  
 width\_cm = float(custom\_width)  
 custom\_size\_format = f"Custom Size: Height: {height\_cm} cm, Width: {width\_cm} cm"  
 filters.append(projects\_table.c.paper\_size.ilike(f"{custom\_size\_format}%"))  
 except ValueError:  
 error = 'Custom height and width must be valid numbers.'  
 elif paper\_size != 'custom':  
 filters.append(projects\_table.c.paper\_size.ilike(f"{paper\_size}%"))  
 elif paper\_size == 'custom' and (not custom\_height or not custom\_width):  
 error = 'Please enter both height and width for custom size.'  
 scale = request.form.get('scale', '').strip()  
 if scale:  
 try:  
 scale\_val = float(scale)  
 # Filter projects by checking if \*any\* associated area has this scale  
 join\_areas = True # Ensure join is active if scale is filtered  
 filters.append(areas\_table.c.scale == scale\_val)  
 except ValueError:  
 error = 'Scale must be a number.'  
  
 # Parse date range  
 date\_from = request.form.get('date\_from', '').strip()  
 date\_to = request.form.get('date\_to', '').strip()  
  
 if date\_from or date\_to:  
 # Convert DD/MM/YYYY format to database format (DD-MM-YY) for comparison  
 def convert\_date\_to\_db\_format(date\_str):  
 try:  
 if date\_str and '/' in date\_str: # DD/MM/YYYY format  
 day, month, year = date\_str.split('/')  
 # Convert to DD-MM-YY format for database comparison  
 return f"{day.zfill(2)}-{month.zfill(2)}-{year[2:]}"  
 elif date\_str and '-' in date\_str: # DD-MM-YY format (already correct)  
 return date\_str  
 return None  
 except:  
 return None  
  
 if date\_from:  
 converted\_from = convert\_date\_to\_db\_format(date\_from)  
 if converted\_from:  
 # For date comparison, we need to ensure proper string comparison  
 filters.append(projects\_table.c.date >= converted\_from)  
 else:  
 error = 'Invalid date format for "From Date". Use DD/MM/YYYY format.'  
  
 if date\_to:  
 converted\_to = convert\_date\_to\_db\_format(date\_to)  
 if converted\_to:  
 # For date comparison, we need to ensure proper string comparison  
 filters.append(projects\_table.c.date <= converted\_to)  
 else:  
 error = 'Invalid date format for "To Date". Use DD/MM/YYYY format.'  
  
 # Parse intersection range filter  
 intersection\_range\_enabled = request.form.get('relative\_size') == '1'  
 intersection\_range\_from = request.form.get('relative\_size\_from', '').strip()  
 intersection\_range\_to = request.form.get('relative\_size\_to', '').strip()  
  
 # Validation: if intersection range is enabled, both values must be provided and valid  
 if intersection\_range\_enabled:  
 if not intersection\_range\_from or not intersection\_range\_to:  
 error = 'Please enter both "From" and "To" values for Intersection Range.'  
 else:  
 try:  
 float(intersection\_range\_from)  
 float(intersection\_range\_to)  
 except ValueError:  
 error = 'Intersection range values must be valid numbers.'  
  
 if error is None:  
 with engine.connect() as conn:  
 # Always join areas to retrieve scales if we're going to display them  
 # For search results, we want to show all scales for a project if multiple exist  
 join\_stmt = projects\_table.join(areas\_table, projects\_table.c.uuid == areas\_table.c.project\_id, isouter=True)  
  
 # Remove all references to relative\_size, inside, outside, percentage\_overlap, and their post-processing  
 # The percentage\_overlap\_enabled and overlap\_percentage logic is removed.  
 # The default spatial filter is now always INSIDE.  
  
 # Combine all spatial filters with OR operator  
 if filters:  
 results = conn.execute(select(\*projects\_table.c, \*areas\_table.c).select\_from(join\_stmt).where(and\_(\*filters))).fetchall()  
  
 # Apply intersection range filter if enabled  
 if intersection\_range\_enabled and bottom\_left and top\_right and intersection\_range\_from and intersection\_range\_to:  
 try:  
 intersection\_from = float(intersection\_range\_from)  
 intersection\_to = float(intersection\_range\_to)  
 required\_area = calculate\_area\_size(xmin, ymin, xmax, ymax)  
 filtered\_results = []  
 for res in results:  
 res\_dict = row\_to\_dict(res)  
 # Only filter if area coordinates are present  
 if all(res\_dict.get(k) is not None for k in ['xmin', 'ymin', 'xmax', 'ymax']):  
 # Calculate intersection area  
 intersect\_xmin = max(res\_dict['xmin'], xmin)  
 intersect\_ymin = max(res\_dict['ymin'], ymin)  
 intersect\_xmax = min(res\_dict['xmax'], xmax)  
 intersect\_ymax = min(res\_dict['ymax'], ymax)  
 if intersect\_xmin < intersect\_xmax and intersect\_ymin < intersect\_ymax:  
 intersection\_area = (intersect\_xmax - intersect\_xmin) \* (intersect\_ymax - intersect\_ymin)  
 intersection\_pct = (intersection\_area / required\_area) \* 100 if required\_area > 0 else 0  
 if intersection\_from <= intersection\_pct <= intersection\_to:  
 filtered\_results.append(res\_dict)  
 else:  
 # If area coordinates are missing, skip filtering  
 filtered\_results.append(res\_dict)  
 results = filtered\_results  
 except ValueError:  
 error = 'Intersection range values must be valid numbers.'  
  
 # Get the filtered projects and their associated scales  
 processed\_results = []  
 for res in results:  
 res\_dict = row\_to\_dict(res)  
 uuid = res\_dict['uuid']  
 # The project\_scales logic is removed as percentage overlap is gone.  
 # If you want to show associated\_scales, use the value from the query or set to None.  
 res\_dict['associated\_scales'] = res\_dict.get('associated\_scales', None)  
 processed\_results.append(res\_dict)  
 results = processed\_results  
 else:  
 # Standard filtering - we need to group by project to get all scales per project  
 sel = select(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description, # <-- Added  
 func.group\_concat(distinct(areas\_table.c.scale)).label('associated\_scales') # Aggregate scales  
 ).select\_from(join\_stmt)  
  
 if filters:  
 sel = sel.where(and\_(\*filters))  
  
 sel = sel.group\_by(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description # <-- Added  
 )  
 results = [row.\_mapping for row in conn.execute(sel)]  
  
 # Add absolute file location for file explorer links  
 processed\_results = []  
 for row in results or []:  
 proj = row\_to\_dict(row)  
 rel\_path = proj['file\_location']  
 abs\_path = os.path.abspath(rel\_path)  
 proj['abs\_file\_location'] = abs\_path  
 proj['abs\_file\_location\_url'] = abs\_path.replace("\\", "/")  
  
 file\_types = [('pdf', 'pdf'), ('jpeg', 'img'), ('jpg', 'img'), ('png', 'img')]  
 all\_files = []  
 most\_recent = None  
  
 for ext, ftype in file\_types:  
 pattern = os.path.join(abs\_path, f"\*.{ext}")  
 files = glob2.glob(pattern)  
 for f in files:  
 ctime = os.path.getctime(f)  
 file\_info = {  
 'path': f,  
 'type': ftype,  
 'ctime': ctime,  
 'filename': os.path.basename(f),  
 'rel\_path': os.path.relpath(f, os.path.abspath('.'))  
 }  
 all\_files.append(file\_info)  
  
 if (most\_recent is None) or (ctime > most\_recent['ctime']):  
 most\_recent = file\_info  
  
 all\_files.sort(key=lambda x: x['ctime'], reverse=True)  
 proj['all\_files'] = all\_files  
 proj['file\_count'] = len(all\_files)  
  
 if most\_recent:  
 proj['view\_file\_path'] = os.path.relpath(most\_recent['path'], PROJECT\_ROOT)  
 proj['view\_file\_type'] = most\_recent['type']  
 else:  
 proj['view\_file\_path'] = None  
 proj['view\_file\_type'] = None  
  
 processed\_results.append(proj)  
  
 results = processed\_results  
 # This block handles GET requests for pagination and table filters  
 # For "All Projects" table  
 projects\_current\_page = request.args.get('page', 1, type=int)  
 projects\_per\_page = request.args.get('per\_page', 10, type=int)  
  
 projects\_filters = {  
 'uuid\_filter': request.args.get('projects\_uuid\_filter', '', type=str),  
 'project\_name\_filter': request.args.get('projects\_project\_name\_filter', '', type=str),  
 'user\_name\_filter': request.args.get('projects\_user\_name\_filter', '', type=str),  
 'date\_filter': request.args.get('projects\_date\_filter', '', type=str),  
 'date\_from\_filter': request.args.get('projects\_date\_from\_filter', '', type=str),  
 'date\_to\_filter': request.args.get('projects\_date\_to\_filter', '', type=str),  
 'file\_location\_filter': request.args.get('projects\_file\_location\_filter', '', type=str),  
 'paper\_size\_filter': request.args.get('projects\_paper\_size\_filter', '', type=str),  
 'associated\_scales\_filter': request.args.get('projects\_associated\_scales\_filter', '', type=str) # New filter  
 }  
  
 projects\_query\_filters = []  
 if projects\_filters['uuid\_filter']:  
 projects\_query\_filters.append(projects\_table.c.uuid.ilike(f"{projects\_filters['uuid\_filter']}%"))  
 if projects\_filters['project\_name\_filter']:  
 projects\_query\_filters.append(projects\_table.c.project\_name.ilike(f"{projects\_filters['project\_name\_filter']}%"))  
 if projects\_filters['user\_name\_filter']:  
 projects\_query\_filters.append(projects\_table.c.user\_name.ilike(f"{projects\_filters['user\_name\_filter']}%"))  
 if projects\_filters['date\_filter']:  
 projects\_query\_filters.append(projects\_table.c.date.ilike(f"{projects\_filters['date\_filter']}%"))  
 if projects\_filters['file\_location\_filter']:  
 projects\_query\_filters.append(projects\_table.c.file\_location.ilike(f"{projects\_filters['file\_location\_filter']}%"))  
 if projects\_filters['paper\_size\_filter']:  
 projects\_query\_filters.append(projects\_table.c.paper\_size.ilike(f"{projects\_filters['paper\_size\_filter']}%"))  
 if projects\_filters['associated\_scales\_filter']:  
 # This filter needs to apply to the aggregated 'associated\_scales' string  
 # It's more complex as it's not a direct column. We'll handle this in the main query.  
 pass  
  
 # For "All Areas" table  
 areas\_current\_page = request.args.get('areas\_page', 1, type=int)  
 areas\_per\_page = request.args.get('areas\_per\_page', 10, type=int)  
  
 areas\_filters = {  
 'id\_filter': request.args.get('areas\_id\_filter', '', type=str),  
 'project\_id\_filter': request.args.get('areas\_project\_id\_filter', '', type=str),  
 'xmin\_filter': request.args.get('areas\_xmin\_filter', '', type=str),  
 'ymin\_filter': request.args.get('areas\_ymin\_filter', '', type=str),  
 'xmax\_filter': request.args.get('areas\_xmax\_filter', '', type=str),  
 'ymax\_filter': request.args.get('areas\_ymax\_filter', '', type=str),  
 'scale\_filter': request.args.get('areas\_scale\_filter', '', type=str),  
 }  
  
 areas\_query\_filters = []  
 if areas\_filters['id\_filter']:  
 try:  
 id\_val = int(areas\_filters['id\_filter'])  
 areas\_query\_filters.append(areas\_table.c.id == id\_val)  
 except ValueError:  
 areas\_query\_filters.append(areas\_table.c.id == -1)  
 if areas\_filters['project\_id\_filter']:  
 areas\_query\_filters.append(areas\_table.c.project\_id.ilike(f"%{areas\_filters['project\_id\_filter']}%"))  
 if areas\_filters['xmin\_filter']:  
 try:  
 xmin\_val = float(areas\_filters['xmin\_filter'])  
 areas\_query\_filters.append(areas\_table.c.xmin == xmin\_val)  
 except ValueError:  
 areas\_query\_filters.append(areas\_table.c.xmin == -1)  
 if areas\_filters['ymin\_filter']:  
 try:  
 ymin\_val = float(areas\_filters['ymin\_filter'])  
 areas\_query\_filters.append(areas\_table.c.ymin == ymin\_val)  
 except ValueError:  
 areas\_query\_filters.append(areas\_table.c.ymin == -1)  
 if areas\_filters['xmax\_filter']:  
 try:  
 xmax\_val = float(areas\_filters['xmax\_filter'])  
 areas\_query\_filters.append(areas\_table.c.xmax == xmax\_val)  
 except ValueError:  
 areas\_query\_filters.append(areas\_table.c.xmax == -1)  
 if areas\_filters['ymax\_filter']:  
 try:  
 ymax\_val = float(areas\_filters['ymax\_filter'])  
 areas\_query\_filters.append(areas\_table.c.ymax == ymax\_val)  
 except ValueError:  
 areas\_query\_filters.append(areas\_table.c.ymax == -1)  
 if areas\_filters['scale\_filter']:  
 try:  
 scale\_val = float(areas\_filters['scale\_filter'])  
 areas\_query\_filters.append(areas\_table.c.scale == scale\_val)  
 except ValueError:  
 areas\_query\_filters.append(areas\_table.c.scale == -1)  
  
  
 with engine.connect() as conn:  
 # For "All Projects" table: Join projects and areas, group by project, and aggregate scales  
 projects\_join\_stmt = projects\_table.outerjoin(areas\_table, projects\_table.c.uuid == areas\_table.c.project\_id)  
  
 # Base query for projects with aggregated scales  
 projects\_base\_query = select(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description, # <-- Added  
 func.group\_concat(distinct(areas\_table.c.scale)).label('associated\_scales')  
 ).select\_from(projects\_join\_stmt).group\_by(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description # <-- Added  
 )  
  
 # Apply basic filters directly  
 for f in projects\_query\_filters:  
 projects\_base\_query = projects\_base\_query.where(f)  
  
 # If there's a filter for associated\_scales, it needs to be applied after aggregation  
 # This requires subquerying or applying a HAVING clause, which SQLAlchemy's `label` helps with.  
 if projects\_filters['associated\_scales\_filter']:  
 scale\_filter\_val = projects\_filters['associated\_scales\_filter']  
 # Convert float to string for comparison with concatenated string  
 projects\_base\_query = projects\_base\_query.having(  
 func.group\_concat(distinct(areas\_table.c.scale)).like(f"%{scale\_filter\_val}%")  
 )  
  
  
 # Get total count for projects pagination  
 # This needs to be done carefully when using group\_by.  
 # A subquery is usually the safest way to count distinct projects after filtering and grouping.  
 count\_subquery = select(projects\_table.c.uuid).select\_from(projects\_join\_stmt)  
 for f in projects\_query\_filters:  
 count\_subquery = count\_subquery.where(f)  
 count\_subquery = count\_subquery.group\_by(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description # <-- Added  
 )  
 if projects\_filters['associated\_scales\_filter']:  
 scale\_filter\_val = projects\_filters['associated\_scales\_filter']  
 count\_subquery = count\_subquery.having(  
 func.group\_concat(distinct(areas\_table.c.scale)).like(f"%{scale\_filter\_val}%")  
 )  
  
 projects\_total\_items = conn.execute(select(func.count()).select\_from(count\_subquery.subquery())).scalar\_one()  
  
 projects\_total\_pages = (projects\_total\_items + projects\_per\_page - 1) // projects\_per\_page  
 if projects\_current\_page > projects\_total\_pages and projects\_total\_pages > 0:  
 projects\_current\_page = projects\_total\_pages  
 elif projects\_total\_pages == 0:  
 projects\_current\_page = 1 # No pages if no items  
  
 # Query projects for the current page with filters and pagination  
 projects\_stmt = projects\_base\_query.limit(projects\_per\_page).offset((projects\_current\_page - 1) \* projects\_per\_page)  
 projects = conn.execute(projects\_stmt).fetchall()  
  
 # Add file information for projects (same as in search results)  
 projects\_list = []  
 for proj in projects:  
 proj\_dict = row\_to\_dict(proj)  
 rel\_path = proj\_dict['file\_location']  
 abs\_path = os.path.abspath(rel\_path)  
 proj\_dict['abs\_file\_location'] = abs\_path  
 proj\_dict['abs\_file\_location\_url'] = abs\_path.replace("\\", "/")  
  
 # Find all files (PDF, JPEG, PNG) for this project  
 file\_types = [('pdf', 'pdf'), ('jpeg', 'img'), ('jpg', 'img'), ('png', 'img')]  
 all\_files = []  
 most\_recent = None  
  
 for ext, ftype in file\_types:  
 pattern = os.path.join(abs\_path, f"\*.{ext}")  
 files = glob2.glob(pattern)  
 for f in files:  
 ctime = os.path.getctime(f)  
 file\_info = {  
 'path': f,  
 'type': ftype,  
 'ctime': ctime,  
 'filename': os.path.basename(f),  
 'rel\_path': os.path.relpath(f, os.path.abspath('.'))  
 }  
 all\_files.append(file\_info)  
  
 # Track the most recent file for the single "View" option  
 if (most\_recent is None) or (ctime > most\_recent['ctime']):  
 most\_recent = file\_info  
  
 # Sort files by creation time (newest first)  
 all\_files.sort(key=lambda x: x['ctime'], reverse=True)  
 proj\_dict['all\_files'] = all\_files  
 proj\_dict['file\_count'] = len(all\_files)  
  
 if most\_recent:  
 proj\_dict['view\_file\_path'] = os.path.relpath(most\_recent['path'], PROJECT\_ROOT)  
 proj\_dict['view\_file\_type'] = most\_recent['type']  
 else:  
 proj\_dict['view\_file\_path'] = None  
 proj\_dict['view\_file\_type'] = None  
  
 projects\_list.append(proj\_dict)  
  
 projects = projects\_list # Replace the original list with the processed one  
  
 # Get total count for areas pagination  
 areas\_count\_stmt = select(func.count()).select\_from(areas\_table)  
 if areas\_query\_filters:  
 areas\_count\_stmt = areas\_count\_stmt.where(and\_(\*areas\_query\_filters))  
 areas\_total\_items = conn.execute(areas\_count\_stmt).scalar\_one()  
  
 areas\_total\_pages = (areas\_total\_items + areas\_per\_page - 1) // areas\_per\_page  
 if areas\_current\_page > areas\_total\_pages and areas\_total\_pages > 0:  
 areas\_current\_page = areas\_total\_pages  
 elif areas\_total\_pages == 0:  
 areas\_current\_page = 1 # No pages if no items  
  
 # Query areas for the current page with filters, joined with projects to get file location  
 areas\_stmt = select(areas\_table.c.id, areas\_table.c.project\_id, areas\_table.c.xmin, areas\_table.c.ymin, areas\_table.c.xmax, areas\_table.c.ymax, areas\_table.c.scale, projects\_table.c.file\_location.label('project\_file\_location'))  
 areas\_stmt = areas\_stmt.select\_from(areas\_table.join(projects\_table, areas\_table.c.project\_id == projects\_table.c.uuid))  
 if areas\_query\_filters:  
 areas\_stmt = areas\_stmt.where(and\_(\*areas\_query\_filters))  
 areas\_stmt = areas\_stmt.limit(areas\_per\_page).offset((areas\_current\_page - 1) \* areas\_per\_page)  
 areas = conn.execute(areas\_stmt).fetchall()  
  
 # Add file information for areas (show files of associated project)  
 areas\_list = []  
 for area in areas:  
 area\_dict = row\_to\_dict(area)  
 project\_file\_location = area\_dict['project\_file\_location']  
 abs\_path = os.path.abspath(project\_file\_location)  
 area\_dict['project\_abs\_file\_location'] = abs\_path  
  
 # Find all files (PDF, JPEG, PNG) for the associated project  
 file\_types = [('pdf', 'pdf'), ('jpeg', 'img'), ('jpg', 'img'), ('png', 'img')]  
 all\_files = []  
 most\_recent = None  
  
 for ext, ftype in file\_types:  
 pattern = os.path.join(abs\_path, f"\*.{ext}")  
 files = glob2.glob(pattern)  
 for f in files:  
 ctime = os.path.getctime(f)  
 file\_info = {  
 'path': f,  
 'type': ftype,  
 'ctime': ctime,  
 'filename': os.path.basename(f),  
 'rel\_path': os.path.relpath(f, os.path.abspath('.'))  
 }  
 all\_files.append(file\_info)  
  
 # Track the most recent file for the single "View" option  
 if (most\_recent is None) or (ctime > most\_recent['ctime']):  
 most\_recent = file\_info  
  
 # Sort files by creation time (newest first)  
 all\_files.sort(key=lambda x: x['ctime'], reverse=True)  
 area\_dict['project\_all\_files'] = all\_files  
 area\_dict['project\_file\_count'] = len(all\_files)  
  
 if most\_recent:  
 area\_dict['project\_view\_file\_path'] = most\_recent['rel\_path']  
 area\_dict['project\_view\_file\_type'] = most\_recent['type']  
 else:  
 area\_dict['project\_view\_file\_path'] = None  
 area\_dict['project\_view\_file\_type'] = None  
  
 areas\_list.append(area\_dict)  
  
 areas = areas\_list # Replace the original list with the processed one  
  
  
 return render\_template(  
 'index.html',  
 results=results,  
 error=error,  
 projects=projects,  
 areas=areas,  
 user\_names=user\_names,  
 selected\_user\_names=selected\_user\_names,  
 projects\_current\_page=projects\_current\_page,  
 projects\_per\_page=projects\_per\_page,  
 projects\_total\_pages=projects\_total\_pages,  
 projects\_filters=projects\_filters,  
 areas\_current\_page=areas\_current\_page,  
 areas\_per\_page=areas\_per\_page,  
 areas\_total\_pages=areas\_total\_pages,  
 areas\_filters=areas\_filters,  
 request=request # Pass request object to access form values for sticky inputs  
 )  
  
@app.route('/view\_file/<path:rel\_path>')  
def view\_file(rel\_path):  
 import os  
 abs\_path = os.path.abspath(os.path.join(PROJECT\_ROOT, rel\_path))  
 print(f"Requested: {abs\_path}")  
 print(f"Project root: {PROJECT\_ROOT}")  
 print(f"Startswith: {abs\_path.startswith(PROJECT\_ROOT)}")  
 # Security: Only allow files inside your project directory  
 if not abs\_path.startswith(PROJECT\_ROOT):  
 return "Access denied", 403  
 return send\_file(abs\_path)  
  
@app.route('/delete\_project/<uuid>', methods=['POST'])  
def delete\_project(uuid):  
 import shutil  
 # Use engine.begin() for a transaction that auto-commits  
 with engine.begin() as conn:  
 # Get the file location for this project  
 sel = select(projects\_table.c.file\_location).where(projects\_table.c.uuid == uuid)  
 result = conn.execute(sel).first()  
 print(f"[DEBUG] Deletion requested for UUID: {uuid}")  
 if result and result[0]:  
 folder = result[0]  
 print(f"[DEBUG] Project folder to delete: {folder}")  
 if os.path.exists(folder) and os.path.isdir(folder):  
 try:  
 shutil.rmtree(folder)  
 print(f"[DEBUG] Folder deleted: {folder}")  
 except Exception as e:  
 print(f"[DEBUG] Error deleting folder: {e}")  
 else:  
 print(f"[DEBUG] Folder does not exist or is not a directory: {folder}")  
 proj\_result = conn.execute(projects\_table.delete().where(projects\_table.c.uuid == uuid))  
 print(f"[DEBUG] Projects deleted: {proj\_result.rowcount}")  
 area\_result = conn.execute(areas\_table.delete().where(areas\_table.c.project\_id == uuid))  
 print(f"[DEBUG] Areas deleted: {area\_result.rowcount}")  
 print(f"[DEBUG] Deletion complete for UUID: {uuid}")  
 return redirect(url\_for('index'))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 try:  
 from config import FLASK\_HOST, FLASK\_DEBUG  
 app.run(host=FLASK\_HOST, port=5002, debug=FLASK\_DEBUG)  
 except ImportError:  
 # Fallback if config file doesn't exist  
 app.run(host='0.0.0.0', port=5002, debug=False)

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\config.py

# Configuration file for ArcSpatialDB  
# Update these values according to your deployment environment  
  
# API Configuration  
API\_BASE\_URL = "http://localhost:5000" # Local Flask app  
API\_TIMEOUT = 30 # Timeout in seconds for API requests  
  
# Database Configuration (for local fallback)  
LOCAL\_DATABASE\_PATH = "elements.db"  
  
# Flask App Configuration  
FLASK\_HOST = "0.0.0.0" # Allow external connections  
FLASK\_PORT = 5000  
FLASK\_DEBUG = False # Set to False in production  
  
# File Upload Configuration  
UPLOAD\_FOLDER = "sampleDataset"  
MAX\_CONTENT\_LENGTH = 16 \* 1024 \* 1024 # 16MB max file size

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\FILTERING\_TEST\_GUIDE.bat

@echo off  
echo ========================================  
echo 🧪 FILTERING TEST GUIDE 🧪  
echo ========================================  
echo.  
  
echo The Node.js backend filtering IS working correctly!  
echo Here's how to test it:  
echo.  
  
echo 1. Test via Browser:  
echo - Open: http://localhost:8000  
echo - In the Projects table, find the UUID filter box  
echo - Enter a partial UUID (e.g., first 4 characters)  
echo - Press Enter or click outside the box  
echo - Results should filter immediately  
echo.  
  
echo 2. Test via API directly:  
echo.  
  
echo Testing API filtering now...  
echo.  
  
echo Testing without filters:  
powershell -Command "$r = Invoke-WebRequest 'http://localhost:5000/api/projects?page=1&per\_page=3' -UseBasicParsing; $j = $r.Content | ConvertFrom-Json; Write-Host \"Found: $($j.projects.Count) projects\"; $firstUuid = $j.projects[0].uuid; $testFilter = $firstUuid.Substring(0,4); Write-Host \"Sample UUID: $firstUuid\"; Write-Host \"Testing filter: $testFilter\"; Write-Host ''; Write-Host 'Testing WITH UUID filter:'; $r2 = Invoke-WebRequest \"http://localhost:5000/api/projects?page=1&per\_page=10&uuid\_filter=$testFilter\" -UseBasicParsing; $j2 = $r2.Content | ConvertFrom-Json; Write-Host \"Filtered results: $($j2.projects.Count) projects\"; if ($j2.projects.Count -lt $j.projects.Count) { Write-Host '✅ Filtering WORKS!' -ForegroundColor Green } else { Write-Host '❌ Filtering issue' -ForegroundColor Red }"  
  
echo.  
echo ========================================  
echo 🎯 TROUBLESHOOTING TIPS  
echo ========================================  
echo.  
echo If filtering doesn't work in the frontend:  
echo.  
echo 1. Check browser Developer Tools (F12)  
echo - Look for JavaScript errors in Console  
echo - Check Network tab for API requests  
echo.  
echo 2. Verify filter input:  
echo - Make sure you're typing in the filter box  
echo - Press Enter or click outside to trigger  
echo.  
echo 3. Clear browser cache:  
echo - Press Ctrl+F5 to hard refresh  
echo.  
echo The Node.js backend filtering is confirmed working!  
echo The issue would be in the frontend interaction.  
echo.  
pause

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\fix\_database.py

import sqlite3  
import os  
  
def fix\_database():  
 """Fix file paths in the database to use relative paths and update some user names"""  
   
 # Connect to the database  
 conn = sqlite3.connect('elements.db')  
 cursor = conn.cursor()  
   
 # Define the mapping for user name changes  
 user\_name\_changes = {  
 'Yoav': {  
 'first': 'Yoav',  
 'second': 'Yoav',   
 'third': 'Yoav',  
 'another\_one': 'Sarah',  
 'last\_one': 'Sarah',  
 'A3\_test': 'Michael',  
 'custom\_size': 'Michael',  
 'CustomSizeCorrect': 'Lisa',  
 'A3\_correct': 'Lisa'  
 }  
 }  
   
 try:  
 # Get all projects  
 cursor.execute("SELECT uuid, project\_name, user\_name, file\_location FROM projects")  
 projects = cursor.fetchall()  
   
 print("Current projects in database:")  
 for project in projects:  
 uuid, project\_name, user\_name, file\_location = project  
 print(f" {uuid}: {project\_name} by {user\_name} at {file\_location}")  
   
 print("\nUpdating projects...")  
   
 # Update each project  
 for project in projects:  
 uuid, project\_name, user\_name, file\_location = project  
   
 # Fix file location to use relative path  
 new\_file\_location = f"sampleDataset/{project\_name}"  
   
 # Update user name based on mapping  
 new\_user\_name = user\_name\_changes.get(user\_name, {}).get(project\_name, user\_name)  
   
 # Update the database  
 cursor.execute("""  
 UPDATE projects   
 SET file\_location = ?, user\_name = ?  
 WHERE uuid = ?  
 """, (new\_file\_location, new\_user\_name, uuid))  
   
 print(f" Updated {project\_name}:")  
 print(f" File location: {file\_location} -> {new\_file\_location}")  
 print(f" User name: {user\_name} -> {new\_user\_name}")  
   
 # Commit the changes  
 conn.commit()  
   
 print("\nVerifying changes...")  
   
 # Verify the changes  
 cursor.execute("SELECT uuid, project\_name, user\_name, file\_location FROM projects")  
 updated\_projects = cursor.fetchall()  
   
 print("Updated projects:")  
 for project in updated\_projects:  
 uuid, project\_name, user\_name, file\_location = project  
 print(f" {uuid}: {project\_name} by {user\_name} at {file\_location}")  
   
 print(f"\nSuccessfully updated {len(projects)} projects!")  
   
 except Exception as e:  
 print(f"Error: {e}")  
 conn.rollback()  
 finally:  
 conn.close()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 fix\_database()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\generate\_sample\_db.py

from sqlalchemy import create\_engine, Column, String, Float, Integer, ForeignKey  
from sqlalchemy.ext.declarative import declarative\_base  
from sqlalchemy.orm import sessionmaker, relationship  
import uuid as uuid\_lib  
  
DATABASE\_URL = 'sqlite:///elements.db'  
Base = declarative\_base()  
engine = create\_engine(DATABASE\_URL, echo=True)  
Session = sessionmaker(bind=engine)  
session = Session()  
  
class Project(Base):  
 \_\_tablename\_\_ = 'projects'  
 uuid = Column(String, primary\_key=True)  
 project\_name = Column(String, nullable=False)  
 user\_name = Column(String, nullable=False)  
 date = Column(String, nullable=False) # ISO format  
 file\_location = Column(String, nullable=False)  
 paper\_size = Column(String, nullable=False)  
 scale = Column(Float, nullable=False)  
 areas = relationship('Area', back\_populates='project', cascade='all, delete-orphan')  
  
class Area(Base):  
 \_\_tablename\_\_ = 'areas'  
 id = Column(Integer, primary\_key=True)  
 project\_id = Column(String, ForeignKey('projects.uuid'), nullable=False)  
 xmin = Column(Float, nullable=False)  
 ymin = Column(Float, nullable=False)  
 xmax = Column(Float, nullable=False)  
 ymax = Column(Float, nullable=False)  
 project = relationship('Project', back\_populates='areas')  
  
Base.metadata.drop\_all(engine)  
Base.metadata.create\_all(engine)  
  
sample\_projects = [  
 {  
 'project\_name': 'ProjectA',  
 'user\_name': 'alice',  
 'date': '2024-06-01',  
 'file\_location': '/projects/a/file1.dwg',  
 'paper\_size': 'A1',  
 'scale': 1.0,  
 'areas': [  
 {'xmin': 732387.35, 'ymin': 3595538.73, 'xmax': 740294.94, 'ymax': 3601127.26}  
 ]  
 },  
 {  
 'project\_name': 'ProjectB',  
 'user\_name': 'bob',  
 'date': '2024-06-02',  
 'file\_location': '/projects/b/file2.dwg',  
 'paper\_size': 'A2',  
 'scale': 0.5,  
 'areas': [  
 {'xmin': 741000.00, 'ymin': 3600000.00, 'xmax': 742000.00, 'ymax': 3602000.00},  
 {'xmin': 732400.57, 'ymin': 3595595.88, 'xmax': 740308.17, 'ymax': 3601184.41}  
 ]  
 },  
 {  
 'project\_name': 'ProjectC',  
 'user\_name': 'carol',  
 'date': '2024-06-03',  
 'file\_location': '/projects/c/file3.dwg',  
 'paper\_size': 'A3',  
 'scale': 2.0,  
 'areas': [  
 {'xmin': 733000.12, 'ymin': 3596000.65, 'xmax': 734000.65, 'ymax': 3599000.12}  
 ]  
 },  
 {  
 'project\_name': 'ProjectD',  
 'user\_name': 'dave',  
 'date': '2024-06-04',  
 'file\_location': '/projects/d/file4.dwg',  
 'paper\_size': 'A0',  
 'scale': 1.5,  
 'areas': [  
 {'xmin': 735000.00, 'ymin': 3598000.00, 'xmax': 736000.00, 'ymax': 3600000.00},  
 {'xmin': 737000.00, 'ymin': 3600500.00, 'xmax': 738000.00, 'ymax': 3601500.00}  
 ]  
 },  
 {  
 'project\_name': 'ProjectE',  
 'user\_name': 'eve',  
 'date': '2024-06-05',  
 'file\_location': '/projects/e/file5.dwg',  
 'paper\_size': 'A4',  
 'scale': 0.75,  
 'areas': [  
 {'xmin': 739000.00, 'ymin': 3601000.00, 'xmax': 740000.00, 'ymax': 3602000.00}  
 ]  
 },  
 {  
 'project\_name': 'ProjectF',  
 'user\_name': 'frank',  
 'date': '2024-06-06',  
 'file\_location': '/projects/f/file6.dwg',  
 'paper\_size': 'A2',  
 'scale': 1.25,  
 'areas': [  
 {'xmin': 740500.00, 'ymin': 3603000.00, 'xmax': 741500.00, 'ymax': 3604000.00},  
 {'xmin': 742000.00, 'ymin': 3605000.00, 'xmax': 743000.00, 'ymax': 3606000.00},  
 {'xmin': 744000.00, 'ymin': 3607000.00, 'xmax': 745000.00, 'ymax': 3608000.00}  
 ]  
 }  
]  
  
for proj in sample\_projects:  
 project = Project(  
 uuid=str(uuid\_lib.uuid4()),  
 project\_name=proj['project\_name'],  
 user\_name=proj['user\_name'],  
 date=proj['date'],  
 file\_location=proj['file\_location'],  
 paper\_size=proj['paper\_size'],  
 scale=proj['scale']  
 )  
 for area in proj['areas']:  
 project.areas.append(Area(\*\*area))  
 session.add(project)  
session.commit()  
  
print('\nDatabase contents:')  
for project in session.query(Project).all():  
 print(f"Project: {project.project\_name}, User: {project.user\_name}, Date: {project.date}, File: {project.file\_location}, Paper Size: {project.paper\_size}, Scale: {project.scale}")  
 for area in project.areas:  
 print(f" Area: id={area.id}, xmin={area.xmin}, ymin={area.ymin}, xmax={area.xmax}, ymax={area.ymax}")  
  
session.close()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\reset\_db.py

from sqlalchemy import create\_engine, MetaData, text  
  
# Connect to the database  
DATABASE\_URL = 'sqlite:///elements.db'  
engine = create\_engine(DATABASE\_URL)  
  
# Reflect existing schema  
metadata = MetaData()  
metadata.reflect(bind=engine)  
  
# Open a connection and transaction  
with engine.connect() as conn:  
 trans = conn.begin()  
 try:  
 # Disable foreign key constraints (SQLite)  
 conn.execute(text("PRAGMA foreign\_keys = OFF;"))  
  
 # Delete all rows from all tables  
 for table in reversed(metadata.sorted\_tables):  
 conn.execute(table.delete())  
  
 # Re-enable foreign key constraints  
 conn.execute(text("PRAGMA foreign\_keys = ON;"))  
 trans.commit()  
 print("✅ All rows deleted from all tables.")  
 except Exception as e:  
 trans.rollback()  
 print(f"❌ Error during deletion: {e}")

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\run\_date\_test.py

if \_\_name\_\_ == "\_\_main\_\_":  
 def convert\_html\_date\_to\_db\_format(html\_date):  
 """Convert HTML date input (YYYY-MM-DD) to database format (DD-MM-YY)"""  
 try:  
 if html\_date and len(html\_date) == 10: # YYYY-MM-DD format from HTML date input  
 year, month, day = html\_date.split('-')  
 # Convert to DD-MM-YY format for database comparison  
 return f"{day.zfill(2)}-{month.zfill(2)}-{year[2:]}"  
 return None  
 except:  
 return None  
  
 # Test cases  
 test\_cases = [  
 ("2025-07-09", "09-07-25"), # July 9th, 2025  
 ("2025-12-25", "25-12-25"), # December 25th, 2025  
 ("2025-01-01", "01-01-25"), # January 1st, 2025  
 ("2025-03-15", "15-03-25"), # March 15th, 2025  
 ]  
  
 print("Testing date conversion:")  
 print("HTML Input (YYYY-MM-DD) -> Database Format (DD-MM-YY)")  
 print("-" \* 50)  
  
 for html\_date, expected in test\_cases:  
 result = convert\_html\_date\_to\_db\_format(html\_date)  
 status = "✓" if result == expected else "✗"  
 print(f"{status} {html\_date} -> {result} (expected: {expected})")  
  
 print("\nDatabase format examples from your data:")  
 print("03-07-25 (July 3rd, 2025)")  
 print("09-07-25 (July 9th, 2025)")  
 print("25-12-25 (December 25th, 2025)")  
   
 print("\nTest completed successfully!")

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\run\_server.py

#!/usr/bin/env python3  
"""  
Smart server runner for ArcSpatialDB  
Automatically chooses the best available server  
"""  
  
import sys  
import os  
  
def check\_waitress():  
 """Check if Waitress is available"""  
 try:  
 import waitress  
 return True  
 except ImportError:  
 return False  
  
def check\_gunicorn():  
 """Check if Gunicorn is available"""  
 try:  
 import gunicorn  
 return True  
 except ImportError:  
 return False  
  
def run\_waitress():  
 """Run with Waitress"""  
 from waitress import serve  
 from app import app  
   
 try:  
 from config import FLASK\_HOST, FLASK\_PORT  
 host = FLASK\_HOST  
 port = FLASK\_PORT  
 except ImportError:  
 host = "0.0.0.0"  
 port = 5000  
   
 print("🚀 Starting ArcSpatialDB with Waitress (Production)")  
 print(f"📍 Host: {host}")  
 print(f"🔌 Port: {port}")  
 print(f"🌐 URL: http://{host}:{port}")  
 print("=" \* 50)  
   
 serve(app, host=host, port=port, threads=4)  
  
def run\_gunicorn():  
 """Run with Gunicorn"""  
 import subprocess  
   
 print("🚀 Starting ArcSpatialDB with Gunicorn (Production)")  
 print("📍 Host: 0.0.0.0")  
 print("🔌 Port: 5000")  
 print("🌐 URL: http://0.0.0.0:5000")  
 print("=" \* 50)  
   
 subprocess.run([  
 sys.executable, "-m", "gunicorn",   
 "-w", "4",   
 "-b", "0.0.0.0:5000",   
 "app:app"  
 ])  
  
def run\_development():  
 """Run with Flask development server"""  
 from app import app  
   
 try:  
 from config import FLASK\_HOST, FLASK\_PORT, FLASK\_DEBUG  
 host = FLASK\_HOST  
 port = FLASK\_PORT  
 debug = FLASK\_DEBUG  
 except ImportError:  
 host = "0.0.0.0"  
 port = 5000  
 debug = False  
   
 print("🚀 Starting ArcSpatialDB with Flask (Development)")  
 print(f"📍 Host: {host}")  
 print(f"🔌 Port: {port}")  
 print(f"🌐 URL: http://{host}:{port}")  
 print("⚠️ WARNING: This is a development server, not suitable for production!")  
 print("=" \* 50)  
   
 app.run(host=host, port=port, debug=debug)  
  
def main():  
 """Choose and run the best available server"""  
   
 print("🔍 Checking available servers...")  
   
 if check\_waitress():  
 print("✅ Waitress found - using production server")  
 run\_waitress()  
 elif check\_gunicorn():  
 print("✅ Gunicorn found - using production server")  
 run\_gunicorn()  
 else:  
 print("⚠️ No production server found - using development server")  
 print("💡 Install Waitress: pip install waitress")  
 print("💡 Or install Gunicorn: pip install gunicorn")  
 run\_development()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\server.py

#!/usr/bin/env python3  
"""  
Production server for ArcSpatialDB using Waitress  
This is the recommended way to run the application on a VM  
"""  
  
from waitress import serve  
from app import app  
import os  
import sys  
  
def main():  
 """Run the production server"""  
   
 # Get configuration  
 try:  
 from config import FLASK\_HOST, FLASK\_PORT  
 host = FLASK\_HOST  
 port = FLASK\_PORT  
 except ImportError:  
 # Fallback configuration  
 host = "0.0.0.0" # Allow external connections  
 port = 5000  
   
 print("🚀 Starting ArcSpatialDB Production Server")  
 print(f"📍 Host: {host}")  
 print(f"🔌 Port: {port}")  
 print(f"🌐 URL: http://{host}:{port}")  
 print("=" \* 50)  
   
 # Start the production server  
 serve(app, host=host, port=port, threads=4)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\start\_node.bat

@echo off  
setlocal EnableDelayedExpansion  
title ArcSpatialDB - Node.js + Frontend Launcher  
color 0A  
  
echo.  
echo ╔══════════════════════════════════════════════════╗  
echo ║ ArcSpatialDB Full Stack ║  
echo ║ Node.js + Frontend ║  
echo ╚══════════════════════════════════════════════════╝  
echo.  
  
REM Check if Node.js is installed  
echo [Step 1/5] Checking Node.js installation...  
node --version >nul 2>&1  
if %errorlevel% neq 0 (  
 echo ❌ ERROR: Node.js is not installed!  
 echo Please install Node.js from: https://nodejs.org/  
 echo.  
 pause  
 exit /b 1  
)  
for /f "tokens=\*" %%i in ('node --version') do set NODE\_VERSION=%%i  
echo ✅ Node.js !NODE\_VERSION! detected  
  
REM Check if Python is installed  
echo.  
echo [Step 2/5] Checking Python installation...  
python --version >nul 2>&1  
if %errorlevel% neq 0 (  
 echo ❌ ERROR: Python is not installed!  
 echo Please install Python from: https://python.org/  
 echo.  
 pause  
 exit /b 1  
)  
for /f "tokens=\*" %%i in ('python --version') do set PYTHON\_VERSION=%%i  
echo ✅ !PYTHON\_VERSION! detected  
  
REM Check if npm dependencies are installed  
echo.  
echo [Step 3/5] Checking Node.js dependencies...  
if not exist "backend\_node\node\_modules" (  
 echo ⚠️ Node.js dependencies not found. Installing...  
 cd backend\_node  
 call npm install  
 if !errorlevel! neq 0 (  
 echo ❌ Failed to install dependencies!  
 pause  
 exit /b 1  
 )  
 cd ..  
 echo ✅ Dependencies installed successfully  
) else (  
 echo ✅ Node.js dependencies found  
)  
  
REM Start Node.js Backend  
echo.  
echo [Step 4/5] Starting Node.js Backend Server...  
echo 🚀 Starting backend on http://localhost:5001  
cd backend\_node  
start "ArcSpatialDB Node.js Backend" /min cmd /c "title ArcSpatialDB Backend ^& echo Backend Server Starting... ^& echo. ^& echo ================================== ^& echo ArcSpatialDB Node.js Backend ^& echo Port: 5001 ^& echo API: http://localhost:5001/api ^& echo ================================== ^& echo. ^& node app.js ^& pause"  
cd ..  
  
REM Wait for backend to start  
echo ⏳ Waiting for backend to initialize...  
timeout /t 4 /nobreak >nul  
  
REM Test backend connection  
echo 🔍 Testing backend connection...  
powershell -Command "try { $response = Invoke-WebRequest -Uri 'http://localhost:5001/api/health' -UseBasicParsing -TimeoutSec 3; if ($response.StatusCode -eq 200) { Write-Host ' ✅ Backend is responding' -ForegroundColor Green } } catch { Write-Host ' ⚠️ Backend may still be starting...' -ForegroundColor Yellow }"  
  
REM Start Frontend  
echo.  
echo [Step 5/5] Starting Frontend Web Server...  
echo 🌐 Starting frontend on http://localhost:8000  
cd frontend  
start "ArcSpatialDB Frontend" /min cmd /c "title ArcSpatialDB Frontend ^& echo Frontend Server Starting... ^& echo. ^& echo ================================== ^& echo ArcSpatialDB Frontend ^& echo Port: 8000 ^& echo URL: http://localhost:8000 ^& echo ================================== ^& echo. ^& python -m http.server 8000"  
cd ..  
  
echo.  
echo ╔══════════════════════════════════════════════════╗  
echo ║ 🎉 LAUNCH COMPLETE! 🎉 ║  
echo ╚══════════════════════════════════════════════════╝  
echo.  
echo 📊 Services Status:  
echo • Node.js Backend: ✅ http://localhost:5001  
echo • Frontend Server: ✅ http://localhost:8000  
echo.  
echo 🔗 Quick Links:  
echo • Application: http://localhost:8000  
echo • API Health: http://localhost:5001/api/health  
echo • Projects API: http://localhost:5001/api/projects  
echo • Areas API: http://localhost:5001/api/areas  
echo.  
echo 💡 Both services are running in minimized windows.  
echo Check your taskbar for "ArcSpatialDB Backend" and "ArcSpatialDB Frontend"  
echo.  
  
REM Ask user if they want to open the application  
choice /c YN /m "Open the application in your browser now? (Y/N)"  
if !errorlevel! equ 1 (  
 echo.  
 echo 🌐 Opening ArcSpatialDB in your default browser...  
 start http://localhost:8000  
 timeout /t 2 /nobreak >nul  
)  
  
echo.  
echo ℹ️ To stop the services:  
echo 1. Close the backend and frontend terminal windows, OR  
echo 2. Press Ctrl+C in their respective windows  
echo.  
echo 📝 Log files and output can be seen in the service windows.  
echo.  
  
pause

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\start\_python.bat

@echo off  
title ArcSpatialDB - Complete System Startup  
color 0F  
echo.  
echo ============================================================  
echo ArcSpatialDB - Complete System Startup  
echo ============================================================  
echo.  
echo This will start both the backend API and frontend servers  
echo.  
echo 1. Backend API Server: http://localhost:5000  
echo 2. Frontend Web App: http://localhost:8000  
echo.  
echo ============================================================  
echo.  
  
echo Step 1: Starting Backend API Server...  
start "ArcSpatialDB Backend" cmd /k "cd /d "%~dp0backend" && START\_BACKEND.bat"  
  
echo Waiting 5 seconds for backend to initialize...  
timeout /t 5 /nobreak > nul  
  
echo Step 2: Starting Frontend Web Server...  
start "ArcSpatialDB Frontend" cmd /k "cd /d "%~dp0frontend" && start\_frontend.bat"  
  
echo Waiting 3 seconds for frontend to initialize...  
timeout /t 3 /nobreak > nul  
  
echo.  
echo ============================================================  
echo 🎉 ArcSpatialDB System Started Successfully!  
echo ============================================================  
echo.  
echo 📡 Backend API: http://localhost:5000  
echo 🎨 Frontend App: http://localhost:8000  
echo.  
echo 💡 Open http://localhost:8000 in your browser to use the app  
echo.  
echo ⚠️ Keep both server windows open while using the system  
echo ❌ Close this window or press any key to finish setup  
echo ============================================================  
echo.  
  
pause

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\test.py

from flask import Flask  
app = Flask(\_\_name\_\_)  
  
@app.route("/")  
def home():  
 return "Hello, browser!"  
  
app.run(debug=True)

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\test\_api.py

#!/usr/bin/env python3  
"""  
Test script for ArcSpatialDB API endpoints  
Run this script to test the API functionality  
"""  
  
import requests  
import json  
import uuid  
from datetime import datetime  
  
# Configuration - update this to your VM's URL  
API\_BASE\_URL = "http://localhost:5000" # Change to your VM URL  
  
def test\_add\_project():  
 """Test adding a project via API"""  
 print("🧪 Testing API: Add Project")  
   
 # Generate test data  
 test\_uuid = str(uuid.uuid4())[:8]  
 test\_project\_name = f"Test Project {datetime.now().strftime('%H:%M:%S')}"  
   
 payload = {  
 "uuid": test\_uuid,  
 "project\_name": test\_project\_name,  
 "user\_name": "test\_user",  
 "date": datetime.now().strftime("%d-%m-%y"),  
 "file\_location": f"sampleDataset/{test\_project\_name}",  
 "paper\_size": "A3 (Portrait)",  
 "description": "Test project created via API",  
 "areas": [  
 {  
 "xmin": 100000,  
 "ymin": 200000,  
 "xmax": 110000,  
 "ymax": 210000,  
 "scale": "Scale: 1:50000"  
 }  
 ]  
 }  
   
 try:  
 response = requests.post(f"{API\_BASE\_URL}/api/add\_project", json=payload, timeout=10)  
   
 print(f"Status Code: {response.status\_code}")  
 print(f"Response: {response.json()}")  
   
 if response.status\_code == 201:  
 print("✅ Project added successfully!")  
 return test\_uuid  
 else:  
 print("❌ Failed to add project")  
 return None  
   
 except requests.exceptions.RequestException as e:  
 print(f"❌ Request failed: {e}")  
 return None  
  
def test\_get\_project(uuid):  
 """Test retrieving a project via API"""  
 if not uuid:  
 print("❌ No UUID provided for get test")  
 return  
   
 print(f"\n🧪 Testing API: Get Project {uuid}")  
   
 try:  
 response = requests.get(f"{API\_BASE\_URL}/api/get\_project/{uuid}", timeout=10)  
   
 print(f"Status Code: {response.status\_code}")  
   
 if response.status\_code == 200:  
 project\_data = response.json()  
 print("✅ Project retrieved successfully!")  
 print(f"Project Name: {project\_data.get('project\_name')}")  
 print(f"User: {project\_data.get('user\_name')}")  
 print(f"Areas: {len(project\_data.get('areas', []))}")  
 else:  
 print(f"❌ Failed to get project: {response.json()}")  
   
 except requests.exceptions.RequestException as e:  
 print(f"❌ Request failed: {e}")  
  
def test\_invalid\_request():  
 """Test API with invalid data"""  
 print("\n🧪 Testing API: Invalid Request")  
   
 # Missing required fields  
 invalid\_payload = {  
 "project\_name": "Invalid Project"  
 # Missing uuid, user\_name, etc.  
 }  
   
 try:  
 response = requests.post(f"{API\_BASE\_URL}/api/add\_project", json=invalid\_payload, timeout=10)  
   
 print(f"Status Code: {response.status\_code}")  
 print(f"Response: {response.json()}")  
   
 if response.status\_code == 400:  
 print("✅ API correctly rejected invalid request!")  
 else:  
 print("❌ API should have rejected invalid request")  
   
 except requests.exceptions.RequestException as e:  
 print(f"❌ Request failed: {e}")  
  
def main():  
 """Run all API tests"""  
 print("🚀 Starting ArcSpatialDB API Tests")  
 print(f"API Base URL: {API\_BASE\_URL}")  
 print("=" \* 50)  
   
 # Test 1: Add project  
 test\_uuid = test\_add\_project()  
   
 # Test 2: Get project  
 test\_get\_project(test\_uuid)  
   
 # Test 3: Invalid request  
 test\_invalid\_request()  
   
 print("\n" + "=" \* 50)  
 print("🏁 API Tests Complete")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\test\_coordinate\_parsing.py

#!/usr/bin/env python3  
"""  
Test script for the enhanced coordinate parsing function.  
Demonstrates support for various separators and coordinate formats.  
"""  
  
def parse\_point(s):  
 """  
 Parse coordinate string with support for various separators and formats.  
 Supports: '/', ',', ':', ';', '|', ' ', '\t', '\\', and combinations  
 Also handles WGS84 format and other coordinate system prefixes  
 Handles complex formats like:  
 - WGS84 UTM 36N 735712 E / 3563829 N  
 - WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N  
   
 Returns: (x, y) if successful, or (None, error\_message) if failed  
 """  
 try:  
 s = str(s).strip()  
   
 # Check for empty or whitespace-only input  
 if not s:  
 return None, "Empty coordinate string provided"  
   
 # Handle complex WGS84 UTM format: "WGS84 UTM 36N 735712 E / 3563829 N"  
 if 'WGS84 UTM' in s.upper():  
 import re  
 # Pattern: WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]  
 utm\_pattern = r'WGS84\s+UTM\s+(\d+[NS])\s+(\d+)\s\*[EW]\s\*/\s\*(\d+)\s\*[NS]'  
 match = re.search(utm\_pattern, s, re.IGNORECASE)  
 if match:  
 try:  
 zone = match.group(1)  
 easting = float(match.group(2))  
 northing = float(match.group(3))  
 return (easting, northing), None  
 except ValueError as e:  
 return None, f"Invalid UTM coordinates in '{s}': {str(e)}"  
 else:  
 return None, f"Invalid WGS84 UTM format. Expected: 'WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]'"  
   
 # Handle complex WGS84 Geographic format: "WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N"  
 if 'WGS84 GEO' in s.upper():  
 import re  
 # Pattern: WGS84 Geo [deg]° [min]' [sec]" [E/W] / [deg]° [min]' [sec]" [N/S]  
 geo\_pattern = r'WGS84\s+GEO\s+(\d+)°\s\*(\d+)\'\s\*([\d.]+)"\s\*[EW]\s\*/\s\*(\d+)°\s\*(\d+)\'\s\*([\d.]+)"\s\*[NS]'  
 match = re.search(geo\_pattern, s, re.IGNORECASE)  
 if match:  
 try:  
 # Convert DMS to decimal degrees  
 lon\_deg, lon\_min, lon\_sec = float(match.group(1)), float(match.group(2)), float(match.group(3))  
 lat\_deg, lat\_min, lat\_sec = float(match.group(4)), float(match.group(5)), float(match.group(6))  
   
 # Check if longitude is East or West  
 if 'W' in s.upper():  
 lon\_deg = -lon\_deg  
 if 'S' in s.upper():  
 lat\_deg = -lat\_deg  
   
 # Convert to decimal degrees  
 lon\_decimal = lon\_deg + (lon\_min / 60) + (lon\_sec / 3600)  
 lat\_decimal = lat\_deg + (lat\_min / 60) + (lat\_sec / 3600)  
   
 return (lon\_decimal, lat\_decimal), None  
 except ValueError as e:  
 return None, f"Invalid geographic coordinates in '{s}': {str(e)}"  
 else:  
 return None, f"Invalid WGS84 Geographic format. Expected: 'WGS84 Geo [deg]° [min]' [sec]\" [E/W] / [deg]° [min]' [sec]\" [N/S]'"  
   
 # Handle simple WGS84 and other coordinate system prefixes  
 if s.upper().startswith(('WGS', 'EPSG', 'UTM', 'GEO', 'PROJ')):  
 # Extract coordinates after the prefix  
 # Look for common patterns like "WGS84: 123.456, 789.012" or "UTM 36N: 123456, 789012"  
 import re  
 # Match coordinates after any prefix  
 coord\_match = re.search(r'[:\s]+([-\d.,\s]+)$', s)  
 if coord\_match:  
 s = coord\_match.group(1).strip()  
 else:  
 return None, f"Invalid coordinate system format. Expected: '[SYSTEM]: [x], [y]' or '[SYSTEM] [x], [y]'"  
   
 # Remove any parentheses, brackets, or quotes  
 s = s.strip('()[]{}"\'\'')  
   
 # Try multiple separators in order of preference  
 separators = ['/', ',', ':', ';', '|', '\\', '\t']  
   
 # First try exact separators  
 for sep in separators:  
 if sep in s:  
 parts = s.split(sep, 1) # Split only on first occurrence  
 if len(parts) == 2:  
 x\_str, y\_str = parts[0].strip(), parts[1].strip()  
 # Try to convert to float  
 try:  
 return (float(x\_str), float(y\_str)), None  
 except ValueError:  
 continue  
   
 # If no separator found, try splitting on whitespace  
 if ' ' in s:  
 parts = s.split()  
 if len(parts) >= 2:  
 try:  
 return (float(parts[0]), float(parts[1])), None  
 except ValueError:  
 pass  
   
 # Try regex pattern for coordinates with optional spaces and various separators  
 import re  
 # Pattern: number, optional spaces, separator, optional spaces, number  
 coord\_pattern = r'([-+]?\d\*\.?\d+)\s\*[\/,:;|\t\\]\s\*([-+]?\d\*\.?\d+)'  
 match = re.search(coord\_pattern, s)  
 if match:  
 try:  
 return (float(match.group(1)), float(match.group(2))), None  
 except ValueError:  
 pass  
   
 # Try pattern for coordinates separated by whitespace  
 space\_pattern = r'([-+]?\d\*\.?\d+)\s+([-+]?\d\*\.?\d+)'  
 match = re.search(space\_pattern, s)  
 if match:  
 try:  
 return (float(match.group(1)), float(match.group(2))), None  
 except ValueError:  
 pass  
   
 # If we get here, no valid format was found  
 return None, f"Invalid coordinate format: '{s}'. Expected formats: 'x,y', 'x/y', 'x:y', 'WGS84 UTM 36N 735712 E / 3563829 N', 'WGS84 Geo 35° 30' 0.11\" E / 32° 11' 9.88\" N', etc."  
 except Exception as e:  
 return None, f"Error parsing coordinates '{s}': {str(e)}"  
  
def test\_coordinate\_parsing():  
 """Test the enhanced coordinate parsing function with various formats."""  
   
 test\_cases = [  
 # Basic separators  
 ("123.456/789.012", "Forward slash"),  
 ("123.456,789.012", "Comma"),  
 ("123.456:789.012", "Colon"),  
 ("123.456;789.012", "Semicolon"),  
 ("123.456|789.012", "Pipe"),  
 ("123.456\\789.012", "Backslash"),  
 ("123.456\t789.012", "Tab"),  
   
 # With spaces  
 ("123.456 / 789.012", "Forward slash with spaces"),  
 ("123.456 , 789.012", "Comma with spaces"),  
 ("123.456 : 789.012", "Colon with spaces"),  
 ("123.456 ; 789.012", "Semicolon with spaces"),  
 ("123.456 | 789.012", "Pipe with spaces"),  
 ("123.456 \\ 789.012", "Backslash with spaces"),  
   
 # Whitespace separated  
 ("123.456 789.012", "Space separated"),  
 ("123.456\t789.012", "Tab separated"),  
   
 # With parentheses and brackets  
 ("(123.456, 789.012)", "Parentheses with comma"),  
 ("[123.456, 789.012]", "Brackets with comma"),  
 ("{123.456, 789.012}", "Braces with comma"),  
 ("(123.456/789.012)", "Parentheses with slash"),  
   
 # With quotes  
 ("\"123.456, 789.012\"", "Double quotes"),  
 ("'123.456, 789.012'", "Single quotes"),  
   
 # Coordinate system prefixes  
 ("WGS84: 123.456, 789.012", "WGS84 prefix with colon"),  
 ("WGS84 123.456, 789.012", "WGS84 prefix with space"),  
 ("UTM 36N: 123456, 789012", "UTM prefix with colon"),  
 ("EPSG:4326 123.456, 789.012", "EPSG prefix"),  
 ("GEO: 123.456, 789.012", "GEO prefix"),  
 ("PROJ: 123.456, 789.012", "PROJ prefix"),  
   
 # Complex WGS84 UTM format (from the image)  
 ("WGS84 UTM 36N 735712 E / 3563829 N", "WGS84 UTM format with zone"),  
 ("WGS84 UTM 36S 735712 E / 3563829 S", "WGS84 UTM format southern hemisphere"),  
 ("WGS84 UTM 36N 735712 W / 3563829 N", "WGS84 UTM format with West"),  
   
 # Complex WGS84 Geographic format (from the image)  
 ("WGS84 Geo 35° 30' 0.11\" E / 32° 11' 9.88\" N", "WGS84 Geo DMS format"),  
 ("WGS84 Geo 35° 30' 0.11\" W / 32° 11' 9.88\" S", "WGS84 Geo DMS format West/South"),  
 ("WGS84 Geo 35° 30' 0\" E / 32° 11' 0\" N", "WGS84 Geo DMS format no seconds"),  
   
 # Negative coordinates  
 ("-123.456, -789.012", "Negative coordinates"),  
 ("-123.456/-789.012", "Negative coordinates with slash"),  
   
 # Integer coordinates  
 ("123, 789", "Integer coordinates"),  
 ("123/789", "Integer coordinates with slash"),  
   
 # Mixed formats  
 ("123.456,789.012", "No spaces"),  
 ("123.456 ,789.012", "Space before comma"),  
 ("123.456, 789.012", "Space after comma"),  
 ("123.456 , 789.012", "Spaces around comma"),  
   
 # Edge cases  
 ("0, 0", "Zero coordinates"),  
 ("0.0, 0.0", "Zero decimal coordinates"),  
 ("123.456789, 789.012345", "High precision"),  
   
 # Invalid cases (should return None)  
 ("invalid", "Invalid string"),  
 ("123.456", "Single number"),  
 ("123.456,", "Incomplete coordinates"),  
 (", 789.012", "Incomplete coordinates"),  
 ("", "Empty string"),  
 (" ", "Whitespace only"),  
 ]  
   
 print("Testing Enhanced Coordinate Parsing Function")  
 print("=" \* 60)  
   
 passed = 0  
 failed = 0  
   
 for test\_input, description in test\_cases:  
 result, error\_msg = parse\_point(test\_input)  
   
 if result is not None:  
 x, y = result  
 print(f"✅ PASS: {description:30} | Input: '{test\_input:20}' | Output: ({x}, {y})")  
 passed += 1  
 else:  
 print(f"❌ FAIL: {description:30} | Input: '{test\_input:20}' | Error: {error\_msg}")  
 failed += 1  
   
 print("\n" + "=" \* 60)  
 print(f"Summary: {passed} passed, {failed} failed")  
 print(f"Success rate: {passed/(passed+failed)\*100:.1f}%")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 test\_coordinate\_parsing()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\user\_info.py

import os  
import platform  
import getpass  
import socket  
import subprocess  
  
def get\_user\_info():  
 info = {}  
  
 # Common Info  
 info['Username'] = getpass.getuser()  
 info['Hostname'] = socket.gethostname()  
 info['IP Address'] = socket.gethostbyname(socket.gethostname())  
 info['Home Directory'] = os.path.expanduser("~")  
  
 # Platform-specific info  
 system = platform.system()  
  
 if system == "Windows":  
 info['Domain'] = os.environ.get("USERDOMAIN", "N/A")  
 info['Full Name'] = get\_windows\_full\_name()  
 info['Shell'] = os.environ.get("ComSpec", "N/A") # typically cmd.exe or powershell  
 else:  
 try:  
 import pwd  
 pw = pwd.getpwnam(info['Username'])  
 info['Full Name'] = pw.pw\_gecos.split(',')[0]  
 info['User ID'] = pw.pw\_uid  
 info['Group ID'] = pw.pw\_gid  
 info['Shell'] = pw.pw\_shell  
 except Exception as e:  
 info['Full Name'] = f"Error: {e}"  
 info['Shell'] = "N/A"  
  
 # Print results  
 print("\n--- User Information ---")  
 for k, v in info.items():  
 print(f"{k}: {v}")  
  
 print("\n--- Environment Variables ---")  
 for k in ['USER', 'USERNAME', 'USERDOMAIN', 'LOGNAME', 'SHELL', 'HOME']:  
 print(f"{k}: {os.environ.get(k, 'N/A')}")  
  
def get\_windows\_full\_name():  
 try:  
 username = os.getlogin()  
 domain = os.environ.get("USERDOMAIN", "")  
 command = f'wmic useraccount where "name=\'{username}\' and domain=\'{domain}\'" get fullname'  
 output = subprocess.check\_output(command, shell=True).decode().splitlines()  
 lines = [line.strip() for line in output if line.strip()]  
 return lines[1] if len(lines) > 1 else "N/A"  
 except Exception as e:  
 return f"Error: {e}"  
  
# Run it  
if \_\_name\_\_ == "\_\_main\_\_":  
 get\_user\_info()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\view\_elements.py

from sqlalchemy import create\_engine, MetaData, Table  
from sqlalchemy.orm import sessionmaker  
  
DATABASE\_URL = 'sqlite:///elements.db'  
engine = create\_engine(DATABASE\_URL)  
Session = sessionmaker(bind=engine)  
session = Session()  
metadata = MetaData()  
  
def print\_table(table, session\_or\_conn, label):  
 print(f'\n{label}:')  
 rows = session\_or\_conn.execute(table.select()).fetchall()  
 if rows:  
 for row in rows:  
 props = ', '.join(f"{col}={getattr(row, col)}" for col in table.columns.keys())  
 print(f" {props}")  
 else:  
 print(f' No {label} found.')  
  
# Reflect tables  
try:  
 elements\_table = Table('elements', metadata, autoload\_with=engine)  
 print\_table(elements\_table, session, 'elements')  
except Exception as e:  
 print('Could not reflect or print elements table:', e)  
  
try:  
 areas\_table = Table('areas', metadata, autoload\_with=engine)  
 print\_table(areas\_table, session, 'areas')  
except Exception as e:  
 print('Could not reflect or print areas table:', e)  
  
try:  
 projects\_table = Table('projects', metadata, autoload\_with=engine)  
 with engine.connect() as conn:  
 print\_table(projects\_table, conn, 'projects')  
except Exception as e:  
 print('Could not reflect or print projects table:', e)  
  
session.close()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\app.py

from flask import Flask, jsonify  
from flask\_cors import CORS  
from api.projects import projects\_bp  
from api.areas import areas\_bp  
from api.files import files\_bp  
import os  
  
def create\_app():  
 app = Flask(\_\_name\_\_)  
   
 # Enable CORS for all domains on all routes  
 CORS(app, origins=["\*"])  
   
 # Register blueprints  
 app.register\_blueprint(projects\_bp, url\_prefix='/api')  
 app.register\_blueprint(areas\_bp, url\_prefix='/api')  
 app.register\_blueprint(files\_bp)  
   
 # Health check endpoint  
 @app.route('/api/health')  
 def health\_check():  
 return jsonify({'status': 'healthy', 'message': 'Backend API is running'})  
   
 # Error handlers  
 @app.errorhandler(404)  
 def not\_found(error):  
 return jsonify({'error': 'Endpoint not found'}), 404  
   
 @app.errorhandler(500)  
 def internal\_error(error):  
 return jsonify({'error': 'Internal server error'}), 500  
   
 return app  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = create\_app()  
   
 try:  
 # Try to import config values  
 import sys  
 sys.path.append('..')  
 from config import FLASK\_HOST, FLASK\_PORT, FLASK\_DEBUG  
 app.run(host=FLASK\_HOST, port=FLASK\_PORT, debug=FLASK\_DEBUG)  
 except ImportError:  
 # Fallback if config file doesn't exist  
 app.run(host='0.0.0.0', port=5000, debug=True)

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\run\_server.py

#!/usr/bin/env python  
"""  
Simple test script to run the backend Flask application  
"""  
import sys  
import os  
  
# Add current directory to path for imports  
sys.path.insert(0, os.path.dirname(os.path.abspath(\_\_file\_\_)))  
  
try:  
 from app import create\_app  
 app = create\_app()  
 print("🚀 Starting ArcSpatialDB Backend API...")  
 print("📡 Server will be available at: http://localhost:5000")  
 print("📚 API endpoints available at: http://localhost:5000/api/")  
 print("💊 Health check: http://localhost:5000/api/health")  
 print("🛑 Press Ctrl+C to stop the server")  
   
 app.run(host='0.0.0.0', port=5000, debug=True)  
   
except ImportError as e:  
 print(f"❌ Import error: {e}")  
 print("Make sure all dependencies are installed:")  
 print("pip install Flask Flask-CORS SQLAlchemy glob2")  
except Exception as e:  
 print(f"❌ Error starting server: {e}")  
 sys.exit(1)

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\START\_BACKEND.bat

@echo off  
title ArcSpatialDB - Backend API Server  
color 0A  
echo.  
echo ========================================  
echo ArcSpatialDB Backend API Server  
echo ========================================  
echo.  
echo Starting Flask backend server...  
echo Server will be available at: http://localhost:5000  
echo API endpoints at: http://localhost:5000/api/  
echo.  
echo Press Ctrl+C to stop the server  
echo ========================================  
echo.  
  
cd /d "%~dp0"  
python app.py  
  
echo.  
echo ========================================  
echo Server stopped. Press any key to exit.  
echo ========================================  
pause > nul

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\start\_server.bat

@echo off  
echo Starting ArcSpatialDB Backend API Server...  
echo Server will be available at: http://localhost:5000  
echo Press Ctrl+C to stop the server  
python app.py  
pause

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\test\_api.py

#!/usr/bin/env python  
"""  
Quick test script to verify the backend API is working  
"""  
import requests  
import json  
  
def test\_api():  
 base\_url = "http://localhost:5000"  
   
 try:  
 # Test health endpoint  
 print("🔍 Testing API Health...")  
 response = requests.get(f"{base\_url}/api/health")  
 print(f"✅ Health Check: {response.status\_code}")  
 print(f"📄 Response: {response.json()}")  
   
 # Test user names endpoint  
 print("\n🔍 Testing User Names...")  
 response = requests.get(f"{base\_url}/api/user\_names")  
 print(f"✅ User Names: {response.status\_code}")  
 data = response.json()  
 print(f"📄 Found {len(data.get('user\_names', []))} user names")  
   
 # Test projects endpoint  
 print("\n🔍 Testing Projects...")  
 response = requests.get(f"{base\_url}/api/projects")  
 print(f"✅ Projects: {response.status\_code}")  
 data = response.json()  
 print(f"📄 Found {len(data.get('projects', []))} projects")  
   
 print("\n🎉 All API endpoints are working!")  
   
 except requests.exceptions.ConnectionError:  
 print("❌ Connection Error: Make sure the backend server is running on port 5000")  
 print(" Run: python app.py (from the backend directory)")  
 except Exception as e:  
 print(f"❌ Error: {e}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 test\_api()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\test\_simple.py

#!/usr/bin/env python  
"""  
Simple manual test using urllib (built-in)  
"""  
import urllib.request  
import urllib.error  
import json  
  
def test\_backend():  
 try:  
 print("🔍 Testing backend API at http://localhost:5000...")  
   
 # Test health endpoint  
 with urllib.request.urlopen('http://localhost:5000/api/health') as response:  
 data = json.loads(response.read().decode())  
 print(f"✅ Health Check Success: {data}")  
   
 # Test user names  
 with urllib.request.urlopen('http://localhost:5000/api/user\_names') as response:  
 data = json.loads(response.read().decode())  
 print(f"✅ User Names Success: Found {len(data.get('user\_names', []))} users")  
   
 print("\n🎉 Backend API is working correctly!")  
 print("📡 You can now refresh your frontend at http://localhost:8000")  
   
 except urllib.error.URLError as e:  
 print(f"❌ Connection Error: {e}")  
 print(" Make sure backend server is running on port 5000")  
 except Exception as e:  
 print(f"❌ Error: {e}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 test\_backend()

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\api\areas.py

from flask import Blueprint, jsonify, request  
from sqlalchemy import select, func, and\_  
from models.database import engine, areas\_table, projects\_table  
from utils.file\_utils import get\_project\_files  
import os  
  
areas\_bp = Blueprint('areas', \_\_name\_\_)  
  
@areas\_bp.route('/areas', methods=['GET'])  
def get\_all\_areas():  
 """Get all areas with pagination and filtering"""  
 try:  
 page = request.args.get('page', 1, type=int)  
 per\_page = request.args.get('per\_page', 10, type=int)  
   
 # Filters  
 filters = {}  
 filters['id\_filter'] = request.args.get('id\_filter', '', type=str)  
 filters['project\_id\_filter'] = request.args.get('project\_id\_filter', '', type=str)  
 filters['xmin\_filter'] = request.args.get('xmin\_filter', '', type=str)  
 filters['ymin\_filter'] = request.args.get('ymin\_filter', '', type=str)  
 filters['xmax\_filter'] = request.args.get('xmax\_filter', '', type=str)  
 filters['ymax\_filter'] = request.args.get('ymax\_filter', '', type=str)  
 filters['scale\_filter'] = request.args.get('scale\_filter', '', type=str)  
   
 query\_filters = []  
 if filters['id\_filter']:  
 try:  
 id\_val = int(filters['id\_filter'])  
 query\_filters.append(areas\_table.c.id == id\_val)  
 except ValueError:  
 query\_filters.append(areas\_table.c.id == -1)  
 if filters['project\_id\_filter']:  
 query\_filters.append(areas\_table.c.project\_id.ilike(f"%{filters['project\_id\_filter']}%"))  
 if filters['xmin\_filter']:  
 try:  
 xmin\_val = float(filters['xmin\_filter'])  
 query\_filters.append(areas\_table.c.xmin == xmin\_val)  
 except ValueError:  
 query\_filters.append(areas\_table.c.xmin == -1)  
 if filters['ymin\_filter']:  
 try:  
 ymin\_val = float(filters['ymin\_filter'])  
 query\_filters.append(areas\_table.c.ymin == ymin\_val)  
 except ValueError:  
 query\_filters.append(areas\_table.c.ymin == -1)  
 if filters['xmax\_filter']:  
 try:  
 xmax\_val = float(filters['xmax\_filter'])  
 query\_filters.append(areas\_table.c.xmax == xmax\_val)  
 except ValueError:  
 query\_filters.append(areas\_table.c.xmax == -1)  
 if filters['ymax\_filter']:  
 try:  
 ymax\_val = float(filters['ymax\_filter'])  
 query\_filters.append(areas\_table.c.ymax == ymax\_val)  
 except ValueError:  
 query\_filters.append(areas\_table.c.ymax == -1)  
 if filters['scale\_filter']:  
 try:  
 scale\_val = float(filters['scale\_filter'])  
 query\_filters.append(areas\_table.c.scale == scale\_val)  
 except ValueError:  
 query\_filters.append(areas\_table.c.scale == -1)  
  
 with engine.connect() as conn:  
 # Get total count for areas pagination  
 count\_stmt = select(func.count()).select\_from(areas\_table)  
 if query\_filters:  
 count\_stmt = count\_stmt.where(and\_(\*query\_filters))  
 total\_items = conn.execute(count\_stmt).scalar\_one()  
  
 total\_pages = (total\_items + per\_page - 1) // per\_page  
 if page > total\_pages and total\_pages > 0:  
 page = total\_pages  
 elif total\_pages == 0:  
 page = 1  
  
 # Query areas for the current page with filters, joined with projects to get file location  
 stmt = select(  
 areas\_table.c.id,   
 areas\_table.c.project\_id,   
 areas\_table.c.xmin,   
 areas\_table.c.ymin,   
 areas\_table.c.xmax,   
 areas\_table.c.ymax,   
 areas\_table.c.scale,   
 projects\_table.c.file\_location.label('project\_file\_location')  
 )  
 stmt = stmt.select\_from(areas\_table.join(projects\_table, areas\_table.c.project\_id == projects\_table.c.uuid))  
   
 if query\_filters:  
 stmt = stmt.where(and\_(\*query\_filters))  
   
 stmt = stmt.limit(per\_page).offset((page - 1) \* per\_page)  
 areas = conn.execute(stmt).fetchall()  
  
 # Add file information for areas (show files of associated project)  
 areas\_list = []  
 for area in areas:  
 area\_dict = dict(area)  
 project\_file\_location = area\_dict['project\_file\_location']  
   
 # Add file information  
 file\_info = get\_project\_files(project\_file\_location)  
 area\_dict['project\_all\_files'] = file\_info['all\_files']  
 area\_dict['project\_file\_count'] = file\_info['file\_count']  
   
 # Add absolute file location  
 abs\_path = os.path.abspath(project\_file\_location)  
 area\_dict['project\_abs\_file\_location'] = abs\_path  
   
 if file\_info['most\_recent']:  
 area\_dict['project\_view\_file\_path'] = file\_info['most\_recent']['rel\_path']  
 area\_dict['project\_view\_file\_type'] = file\_info['most\_recent']['type']  
 else:  
 area\_dict['project\_view\_file\_path'] = None  
 area\_dict['project\_view\_file\_type'] = None  
  
 areas\_list.append(area\_dict)  
  
 return jsonify({  
 'areas': areas\_list,  
 'pagination': {  
 'current\_page': page,  
 'per\_page': per\_page,  
 'total\_pages': total\_pages,  
 'total\_items': total\_items  
 },  
 'filters': filters  
 })  
  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\api\files.py

from flask import Blueprint, send\_file  
import os  
  
PROJECT\_ROOT = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '..', '..'))  
  
files\_bp = Blueprint('files', \_\_name\_\_)  
  
@files\_bp.route('/view\_file/<path:rel\_path>')  
def view\_file(rel\_path):  
 """Serve project files for viewing"""  
 try:  
 abs\_path = os.path.abspath(os.path.join(PROJECT\_ROOT, rel\_path))  
   
 # Security: Only allow files inside project directory  
 if not abs\_path.startswith(PROJECT\_ROOT):  
 return {'error': 'Access denied'}, 403  
   
 if not os.path.exists(abs\_path):  
 return {'error': 'File not found'}, 404  
   
 return send\_file(abs\_path)  
 except Exception as e:  
 return {'error': str(e)}, 500

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\api\projects.py

from flask import Blueprint, jsonify, request  
from sqlalchemy import select, distinct, func, and\_, or\_  
from models.database import engine, projects\_table, areas\_table  
from utils.helpers import parse\_point, calculate\_area\_size, convert\_date\_to\_db\_format  
from utils.file\_utils import get\_project\_files  
import os  
  
projects\_bp = Blueprint('projects', \_\_name\_\_)  
  
@projects\_bp.route('/projects', methods=['GET'])  
def get\_all\_projects():  
 """Get all projects with pagination and filtering"""  
 try:  
 page = request.args.get('page', 1, type=int)  
 per\_page = request.args.get('per\_page', 10, type=int)  
   
 # Filters  
 filters = {}  
 filters['uuid\_filter'] = request.args.get('uuid\_filter', '', type=str)  
 filters['project\_name\_filter'] = request.args.get('project\_name\_filter', '', type=str)  
 filters['user\_name\_filter'] = request.args.get('user\_name\_filter', '', type=str)  
 filters['date\_filter'] = request.args.get('date\_filter', '', type=str)  
 filters['date\_from\_filter'] = request.args.get('date\_from\_filter', '', type=str)  
 filters['date\_to\_filter'] = request.args.get('date\_to\_filter', '', type=str)  
 filters['file\_location\_filter'] = request.args.get('file\_location\_filter', '', type=str)  
 filters['paper\_size\_filter'] = request.args.get('paper\_size\_filter', '', type=str)  
 filters['associated\_scales\_filter'] = request.args.get('associated\_scales\_filter', '', type=str)  
   
 query\_filters = []  
 if filters['uuid\_filter']:  
 query\_filters.append(projects\_table.c.uuid.ilike(f"{filters['uuid\_filter']}%"))  
 if filters['project\_name\_filter']:  
 query\_filters.append(projects\_table.c.project\_name.ilike(f"{filters['project\_name\_filter']}%"))  
 if filters['user\_name\_filter']:  
 query\_filters.append(projects\_table.c.user\_name.ilike(f"{filters['user\_name\_filter']}%"))  
 if filters['date\_filter']:  
 query\_filters.append(projects\_table.c.date.ilike(f"{filters['date\_filter']}%"))  
 if filters['file\_location\_filter']:  
 query\_filters.append(projects\_table.c.file\_location.ilike(f"{filters['file\_location\_filter']}%"))  
 if filters['paper\_size\_filter']:  
 query\_filters.append(projects\_table.c.paper\_size.ilike(f"{filters['paper\_size\_filter']}%"))  
  
 with engine.connect() as conn:  
 # Join projects and areas, group by project, and aggregate scales  
 join\_stmt = projects\_table.outerjoin(areas\_table, projects\_table.c.uuid == areas\_table.c.project\_id)  
  
 # Base query for projects with aggregated scales  
 base\_query = select(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description,  
 func.group\_concat(distinct(areas\_table.c.scale)).label('associated\_scales')  
 ).select\_from(join\_stmt).group\_by(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description  
 )  
  
 # Apply basic filters  
 for f in query\_filters:  
 base\_query = base\_query.where(f)  
  
 # Handle associated scales filter  
 if filters['associated\_scales\_filter']:  
 scale\_filter\_val = filters['associated\_scales\_filter']  
 base\_query = base\_query.having(  
 func.group\_concat(distinct(areas\_table.c.scale)).like(f"%{scale\_filter\_val}%")  
 )  
  
 # Get total count for pagination  
 count\_subquery = select(projects\_table.c.uuid).select\_from(join\_stmt)  
 for f in query\_filters:  
 count\_subquery = count\_subquery.where(f)  
 count\_subquery = count\_subquery.group\_by(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description  
 )  
 if filters['associated\_scales\_filter']:  
 scale\_filter\_val = filters['associated\_scales\_filter']  
 count\_subquery = count\_subquery.having(  
 func.group\_concat(distinct(areas\_table.c.scale)).like(f"%{scale\_filter\_val}%")  
 )  
  
 total\_items = conn.execute(select(func.count()).select\_from(count\_subquery.subquery())).scalar\_one()  
 total\_pages = (total\_items + per\_page - 1) // per\_page  
  
 if page > total\_pages and total\_pages > 0:  
 page = total\_pages  
 elif total\_pages == 0:  
 page = 1  
  
 # Query projects for the current page  
 stmt = base\_query.limit(per\_page).offset((page - 1) \* per\_page)  
 projects = conn.execute(stmt).fetchall()  
  
 # Process projects and add file information  
 projects\_list = []  
 for proj in projects:  
 proj\_dict = dict(proj)  
   
 # Add file information  
 file\_info = get\_project\_files(proj\_dict['file\_location'])  
 proj\_dict.update(file\_info)  
   
 # Add absolute file location  
 abs\_path = os.path.abspath(proj\_dict['file\_location'])  
 proj\_dict['abs\_file\_location'] = abs\_path  
 proj\_dict['abs\_file\_location\_url'] = abs\_path.replace("\\", "/")  
   
 if file\_info['most\_recent']:  
 proj\_dict['view\_file\_path'] = file\_info['most\_recent']['rel\_path']  
 proj\_dict['view\_file\_type'] = file\_info['most\_recent']['type']  
 else:  
 proj\_dict['view\_file\_path'] = None  
 proj\_dict['view\_file\_type'] = None  
  
 projects\_list.append(proj\_dict)  
  
 return jsonify({  
 'projects': projects\_list,  
 'pagination': {  
 'current\_page': page,  
 'per\_page': per\_page,  
 'total\_pages': total\_pages,  
 'total\_items': total\_items  
 },  
 'filters': filters  
 })  
  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500  
  
@projects\_bp.route('/projects/search', methods=['POST'])  
def search\_projects():  
 """Search projects with advanced filtering"""  
 try:  
 data = request.get\_json() or {}  
   
 filters = []  
 join\_areas = False  
  
 # Parse spatial box  
 bottom\_left = data.get('bottom\_left', '').strip()  
 top\_right = data.get('top\_right', '').strip()  
   
 if bottom\_left and top\_right:  
 bl\_result = parse\_point(bottom\_left)  
 tr\_result = parse\_point(top\_right)  
   
 # Check for parsing errors  
 if bl\_result[1] is not None: # Error in bottom\_left  
 return jsonify({'error': f'Bottom Left: {bl\_result[1]}'}), 400  
 elif tr\_result[1] is not None: # Error in top\_right  
 return jsonify({'error': f'Top Right: {tr\_result[1]}'}), 400  
 elif not bl\_result[0] or not tr\_result[0]: # No coordinates returned  
 return jsonify({'error': 'Invalid input format. Please use X/Y or X,Y for both points.'}), 400  
 else:  
 xmin, ymin = bl\_result[0]  
 xmax, ymax = tr\_result[0]  
 if xmin >= xmax or ymin >= ymax:  
 return jsonify({'error': 'Bottom Left must be southwest (smaller X and Y) of Top Right. Please check your input.'}), 400  
   
 join\_areas = True  
 # Default INSIDE spatial filter  
 inside\_filters = [  
 areas\_table.c.xmin >= xmin,  
 areas\_table.c.xmax <= xmax,  
 areas\_table.c.ymin >= ymin,  
 areas\_table.c.ymax <= ymax  
 ]  
 filters.append(and\_(\*inside\_filters))  
  
 # Parse other filters  
 uuid = data.get('uuid', '').strip()  
 if uuid:  
 filters.append(projects\_table.c.uuid.ilike(f"{uuid}%"))  
  
 user\_names = data.get('user\_names', [])  
 if user\_names:  
 filters.append(or\_(\*[projects\_table.c.user\_name.ilike(f"{n}%") for n in user\_names]))  
  
 paper\_size = data.get('paper\_size', '').strip()  
 custom\_height = data.get('custom\_height', '').strip()  
 custom\_width = data.get('custom\_width', '').strip()  
  
 if paper\_size:  
 if paper\_size == 'custom' and custom\_height and custom\_width:  
 try:  
 height\_cm = float(custom\_height)  
 width\_cm = float(custom\_width)  
 custom\_size\_format = f"Custom Size: Height: {height\_cm} cm, Width: {width\_cm} cm"  
 filters.append(projects\_table.c.paper\_size.ilike(f"{custom\_size\_format}%"))  
 except ValueError:  
 return jsonify({'error': 'Custom height and width must be valid numbers.'}), 400  
 elif paper\_size != 'custom':  
 filters.append(projects\_table.c.paper\_size.ilike(f"{paper\_size}%"))  
 elif paper\_size == 'custom' and (not custom\_height or not custom\_width):  
 return jsonify({'error': 'Please enter both height and width for custom size.'}), 400  
  
 scale = data.get('scale', '').strip()  
 if scale:  
 try:  
 scale\_val = float(scale)  
 join\_areas = True  
 filters.append(areas\_table.c.scale == scale\_val)  
 except ValueError:  
 return jsonify({'error': 'Scale must be a number.'}), 400  
  
 # Parse date range  
 date\_from = data.get('date\_from', '').strip()  
 date\_to = data.get('date\_to', '').strip()  
  
 if date\_from:  
 converted\_from = convert\_date\_to\_db\_format(date\_from)  
 if converted\_from:  
 filters.append(projects\_table.c.date >= converted\_from)  
 else:  
 return jsonify({'error': 'Invalid date format for "From Date". Use DD/MM/YYYY format.'}), 400  
  
 if date\_to:  
 converted\_to = convert\_date\_to\_db\_format(date\_to)  
 if converted\_to:  
 filters.append(projects\_table.c.date <= converted\_to)  
 else:  
 return jsonify({'error': 'Invalid date format for "To Date". Use DD/MM/YYYY format.'}), 400  
  
 # Parse intersection range filter  
 intersection\_range\_enabled = data.get('relative\_size', False)  
 intersection\_range\_from = data.get('relative\_size\_from', '').strip()  
 intersection\_range\_to = data.get('relative\_size\_to', '').strip()  
  
 if intersection\_range\_enabled:  
 if not intersection\_range\_from or not intersection\_range\_to:  
 return jsonify({'error': 'Please enter both "From" and "To" values for Intersection Range.'}), 400  
 try:  
 float(intersection\_range\_from)  
 float(intersection\_range\_to)  
 except ValueError:  
 return jsonify({'error': 'Intersection range values must be valid numbers.'}), 400  
  
 with engine.connect() as conn:  
 # Join areas to retrieve scales  
 join\_stmt = projects\_table.join(areas\_table, projects\_table.c.uuid == areas\_table.c.project\_id, isouter=True)  
  
 if filters:  
 results = conn.execute(select(\*projects\_table.c, \*areas\_table.c).select\_from(join\_stmt).where(and\_(\*filters))).fetchall()  
  
 # Apply intersection range filter if enabled  
 if intersection\_range\_enabled and bottom\_left and top\_right and intersection\_range\_from and intersection\_range\_to:  
 try:  
 intersection\_from = float(intersection\_range\_from)  
 intersection\_to = float(intersection\_range\_to)  
 required\_area = calculate\_area\_size(xmin, ymin, xmax, ymax)  
 filtered\_results = []  
   
 for res in results:  
 res\_dict = dict(res)  
 if all(res\_dict.get(k) is not None for k in ['xmin', 'ymin', 'xmax', 'ymax']):  
 # Calculate intersection area  
 intersect\_xmin = max(res\_dict['xmin'], xmin)  
 intersect\_ymin = max(res\_dict['ymin'], ymin)  
 intersect\_xmax = min(res\_dict['xmax'], xmax)  
 intersect\_ymax = min(res\_dict['ymax'], ymax)  
   
 if intersect\_xmin < intersect\_xmax and intersect\_ymin < intersect\_ymax:  
 intersection\_area = (intersect\_xmax - intersect\_xmin) \* (intersect\_ymax - intersect\_ymin)  
 intersection\_pct = (intersection\_area / required\_area) \* 100 if required\_area > 0 else 0  
 if intersection\_from <= intersection\_pct <= intersection\_to:  
 filtered\_results.append(res\_dict)  
 else:  
 filtered\_results.append(res\_dict)  
   
 results = filtered\_results  
 except ValueError:  
 return jsonify({'error': 'Intersection range values must be valid numbers.'}), 400  
 else:  
 # Get all projects with aggregated scales  
 sel = select(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description,  
 func.group\_concat(distinct(areas\_table.c.scale)).label('associated\_scales')  
 ).select\_from(join\_stmt).group\_by(  
 projects\_table.c.uuid,  
 projects\_table.c.project\_name,  
 projects\_table.c.user\_name,  
 projects\_table.c.date,  
 projects\_table.c.file\_location,  
 projects\_table.c.paper\_size,  
 projects\_table.c.description  
 )  
   
 results = [row for row in conn.execute(sel)]  
  
 # Process results and add file information  
 processed\_results = []  
 for row in results or []:  
 proj = dict(row)  
   
 # Add file information  
 file\_info = get\_project\_files(proj['file\_location'])  
 proj.update(file\_info)  
   
 # Add absolute file location  
 abs\_path = os.path.abspath(proj['file\_location'])  
 proj['abs\_file\_location'] = abs\_path  
 proj['abs\_file\_location\_url'] = abs\_path.replace("\\", "/")  
   
 if file\_info['most\_recent']:  
 proj['view\_file\_path'] = file\_info['most\_recent']['rel\_path']  
 proj['view\_file\_type'] = file\_info['most\_recent']['type']  
 else:  
 proj['view\_file\_path'] = None  
 proj['view\_file\_type'] = None  
  
 processed\_results.append(proj)  
  
 return jsonify({'results': processed\_results})  
  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500  
  
@projects\_bp.route('/projects/<uuid>', methods=['GET'])  
def get\_project(uuid):  
 """Get a specific project by UUID"""  
 try:  
 with engine.connect() as conn:  
 # Get project details  
 project\_result = conn.execute(  
 select(projects\_table).where(projects\_table.c.uuid == uuid)  
 ).first()  
   
 if not project\_result:  
 return jsonify({'error': 'Project not found'}), 404  
   
 project\_dict = dict(project\_result)  
   
 # Get associated areas  
 areas\_result = conn.execute(  
 select(areas\_table).where(areas\_table.c.project\_id == uuid)  
 ).fetchall()  
   
 areas\_list = [dict(area) for area in areas\_result]  
 project\_dict['areas'] = areas\_list  
   
 # Add file information  
 file\_info = get\_project\_files(project\_dict['file\_location'])  
 project\_dict.update(file\_info)  
   
 return jsonify(project\_dict)  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500  
  
@projects\_bp.route('/projects/<uuid>/files', methods=['GET'])  
def get\_project\_files\_endpoint(uuid):  
 """Get all files for a specific project"""  
 try:  
 with engine.connect() as conn:  
 # Get project file location  
 project\_result = conn.execute(  
 select(projects\_table.c.file\_location).where(projects\_table.c.uuid == uuid)  
 ).first()  
   
 if not project\_result:  
 return jsonify({'error': 'Project not found'}), 404  
   
 file\_location = project\_result[0]  
 file\_info = get\_project\_files(file\_location)  
   
 return jsonify(file\_info)  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500  
  
@projects\_bp.route('/projects/<uuid>', methods=['DELETE'])  
def delete\_project(uuid):  
 """Delete a project and its associated areas"""  
 try:  
 import shutil  
   
 with engine.begin() as conn:  
 # Get the file location for this project  
 sel = select(projects\_table.c.file\_location).where(projects\_table.c.uuid == uuid)  
 result = conn.execute(sel).first()  
   
 if result and result[0]:  
 folder = result[0]  
 if os.path.exists(folder) and os.path.isdir(folder):  
 try:  
 shutil.rmtree(folder)  
 except Exception as e:  
 print(f"Error deleting folder: {e}")  
   
 # Delete from database  
 proj\_result = conn.execute(projects\_table.delete().where(projects\_table.c.uuid == uuid))  
 area\_result = conn.execute(areas\_table.delete().where(areas\_table.c.project\_id == uuid))  
   
 if proj\_result.rowcount == 0:  
 return jsonify({'error': 'Project not found'}), 404  
   
 return jsonify({  
 'message': 'Project deleted successfully',  
 'projects\_deleted': proj\_result.rowcount,  
 'areas\_deleted': area\_result.rowcount  
 })  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500  
  
@projects\_bp.route('/projects', methods=['POST'])  
def add\_project():  
 """Add a new project"""  
 data = request.get\_json()  
   
 if not data:  
 return jsonify({'error': 'No JSON data provided'}), 400  
   
 required\_fields = ['uuid', 'project\_name', 'user\_name', 'date', 'file\_location', 'paper\_size', 'description']  
 missing\_fields = [f for f in required\_fields if f not in data]  
   
 if missing\_fields:  
 return jsonify({'error': f"Missing fields: {', '.join(missing\_fields)}"}), 400  
   
 try:  
 with engine.begin() as conn:  
 # Insert project  
 conn.execute(projects\_table.insert().values(  
 uuid=data['uuid'],  
 project\_name=data['project\_name'],  
 user\_name=data['user\_name'],  
 date=data['date'],  
 file\_location=data['file\_location'],  
 paper\_size=data['paper\_size'],  
 description=data['description']  
 ))  
   
 # Insert areas if provided  
 if 'areas' in data and isinstance(data['areas'], list):  
 for area\_data in data['areas']:  
 area\_required\_fields = ['xmin', 'ymin', 'xmax', 'ymax', 'scale']  
 area\_missing\_fields = [f for f in area\_required\_fields if f not in area\_data]  
   
 if area\_missing\_fields:  
 return jsonify({'error': f"Missing area fields: {', '.join(area\_missing\_fields)}"}), 400  
   
 conn.execute(areas\_table.insert().values(  
 project\_id=data['uuid'],  
 xmin=area\_data['xmin'],  
 ymin=area\_data['ymin'],  
 xmax=area\_data['xmax'],  
 ymax=area\_data['ymax'],  
 scale=area\_data['scale']  
 ))  
   
 return jsonify({'message': 'Project added successfully', 'uuid': data['uuid']}), 201  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500  
  
@projects\_bp.route('/user\_names', methods=['GET'])  
def get\_user\_names():  
 """Get all unique user names"""  
 try:  
 with engine.connect() as conn:  
 user\_names = [row[0] for row in conn.execute(select(projects\_table.c.user\_name).distinct())]  
 return jsonify({'user\_names': user\_names})  
 except Exception as e:  
 return jsonify({'error': str(e)}), 500

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\api\\_\_init\_\_.py

# Backend API package

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\models\database.py

from sqlalchemy import create\_engine, MetaData, Table  
  
# Database configuration  
import os  
DB\_PATH = os.path.join(os.path.dirname(\_\_file\_\_), '..', '..', 'elements.db')  
DATABASE\_URL = f'sqlite:///{DB\_PATH}'  
engine = create\_engine(DATABASE\_URL)  
metadata = MetaData()  
  
# Reflect tables from existing database  
projects\_table = Table('projects', metadata, autoload\_with=engine)  
areas\_table = Table('areas', metadata, autoload\_with=engine)

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\models\\_\_init\_\_.py

# Database models package

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\utils\file\_utils.py

import os  
import glob2  
from datetime import datetime  
  
PROJECT\_ROOT = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '..', '..'))  
  
def get\_project\_files(file\_location):  
 """Get all files (PDF, JPEG, PNG) for a project and return file information"""  
 abs\_path = os.path.abspath(file\_location)  
 file\_types = [('pdf', 'pdf'), ('jpeg', 'img'), ('jpg', 'img'), ('png', 'img')]  
 all\_files = []  
 most\_recent = None  
  
 for ext, ftype in file\_types:  
 pattern = os.path.join(abs\_path, f"\*.{ext}")  
 files = glob2.glob(pattern)  
 for f in files:  
 ctime = os.path.getctime(f)  
 file\_info = {  
 'path': f,  
 'type': ftype,  
 'ctime': ctime,  
 'filename': os.path.basename(f),  
 'rel\_path': os.path.relpath(f, PROJECT\_ROOT)  
 }  
 all\_files.append(file\_info)  
  
 if (most\_recent is None) or (ctime > most\_recent['ctime']):  
 most\_recent = file\_info  
  
 # Sort files by creation time (newest first)  
 all\_files.sort(key=lambda x: x['ctime'], reverse=True)  
   
 return {  
 'all\_files': all\_files,  
 'file\_count': len(all\_files),  
 'most\_recent': most\_recent  
 }

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\utils\helpers.py

def parse\_point(s):  
 """  
 Parse coordinate string with support for various separators and formats.  
 Supports: '/', ',', ':', ';', '|', ' ', '\t', '\\', and combinations  
 Also handles WGS84 format and other coordinate system prefixes  
 Handles complex formats like:  
 - WGS84 UTM 36N 735712 E / 3563829 N  
 - WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N  
   
 Returns: (x, y) if successful, or (None, error\_message) if failed  
 """  
 import re  
 try:  
 s = str(s).strip()  
   
 # Check for empty or whitespace-only input  
 if not s:  
 return None, "Empty coordinate string provided"  
   
 # Handle complex WGS84 UTM format: "WGS84 UTM 36N 735712 E / 3563829 N"  
 if 'WGS84 UTM' in s.upper():  
 # Pattern: WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]  
 utm\_pattern = r'WGS84\s+UTM\s+(\d+[NS])\s+(\d+)\s\*[EW]\s\*/\s\*(\d+)\s\*[NS]'  
 match = re.search(utm\_pattern, s, re.IGNORECASE)  
 if match:  
 try:  
 zone = match.group(1)  
 easting = float(match.group(2))  
 northing = float(match.group(3))  
 return (easting, northing), None  
 except ValueError as e:  
 return None, f"Invalid UTM coordinates in '{s}': {str(e)}"  
 else:  
 return None, f"Invalid WGS84 UTM format. Expected: 'WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]'"  
   
 # Handle complex WGS84 Geographic format: "WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N"  
 if 'WGS84 GEO' in s.upper():  
 # Pattern: WGS84 Geo [deg]° [min]' [sec]" [E/W] / [deg]° [min]' [sec]" [N/S]  
 geo\_pattern = r'WGS84\s+GEO\s+(\d+)°\s\*(\d+)\'\s\*([\d.]+)"\s\*[EW]\s\*/\s\*(\d+)°\s\*(\d+)\'\s\*([\d.]+)"\s\*[NS]'  
 match = re.search(geo\_pattern, s, re.IGNORECASE)  
 if match:  
 try:  
 # Convert DMS to decimal degrees  
 lon\_deg, lon\_min, lon\_sec = float(match.group(1)), float(match.group(2)), float(match.group(3))  
 lat\_deg, lat\_min, lat\_sec = float(match.group(4)), float(match.group(5)), float(match.group(6))  
   
 # Check if longitude is East or West  
 if 'W' in s.upper():  
 lon\_deg = -lon\_deg  
 if 'S' in s.upper():  
 lat\_deg = -lat\_deg  
   
 # Convert to decimal degrees  
 lon\_decimal = lon\_deg + (lon\_min / 60) + (lon\_sec / 3600)  
 lat\_decimal = lat\_deg + (lat\_min / 60) + (lat\_sec / 3600)  
   
 return (lon\_decimal, lat\_decimal), None  
 except ValueError as e:  
 return None, f"Invalid geographic coordinates in '{s}': {str(e)}"  
 else:  
 return None, f"Invalid WGS84 Geographic format. Expected: 'WGS84 Geo [deg]° [min]' [sec]\" [E/W] / [deg]° [min]' [sec]\" [N/S]'"  
   
 # Handle simple WGS84 and other coordinate system prefixes  
 if s.upper().startswith(('WGS', 'EPSG', 'UTM', 'GEO', 'PROJ')):  
 # Extract coordinates after the prefix  
 # Look for common patterns like "WGS84: 123.456, 789.012" or "UTM 36N: 123456, 789012"  
 # Match coordinates after any prefix  
 coord\_match = re.search(r'[:\s]+([-\d.,\s]+)$', s)  
 if coord\_match:  
 s = coord\_match.group(1).strip()  
 else:  
 return None, f"Invalid coordinate system format. Expected: '[SYSTEM]: [x], [y]' or '[SYSTEM] [x], [y]'"  
   
 # Remove any parentheses, brackets, or quotes  
 s = s.strip('()[]{}"\'\'')  
   
 # Try multiple separators in order of preference  
 separators = ['/', ',', ':', ';', '|', '\\', '\t']  
   
 # First try exact separators  
 for sep in separators:  
 if sep in s:  
 parts = s.split(sep, 1) # Split only on first occurrence  
 if len(parts) == 2:  
 x\_str, y\_str = parts[0].strip(), parts[1].strip()  
 # Try to convert to float  
 try:  
 return (float(x\_str), float(y\_str)), None  
 except ValueError:  
 continue  
   
 # If no separator found, try splitting on whitespace  
 if ' ' in s:  
 parts = s.split()  
 if len(parts) >= 2:  
 try:  
 return (float(parts[0]), float(parts[1])), None  
 except ValueError:  
 pass  
   
 # Try regex pattern for coordinates with optional spaces and various separators  
 # Pattern: number, optional spaces, separator, optional spaces, number  
 coord\_pattern = r'([-+]?\d\*\.?\d+)\s\*[\/,:;|\t\\]\s\*([-+]?\d\*\.?\d+)'  
 match = re.search(coord\_pattern, s)  
 if match:  
 try:  
 return (float(match.group(1)), float(match.group(2))), None  
 except ValueError:  
 pass  
   
 # Try pattern for coordinates separated by whitespace  
 space\_pattern = r'([-+]?\d\*\.?\d+)\s+([-+]?\d\*\.?\d+)'  
 match = re.search(space\_pattern, s)  
 if match:  
 try:  
 return (float(match.group(1)), float(match.group(2))), None  
 except ValueError:  
 pass  
   
 # If we get here, no valid format was found  
 return None, f"Invalid coordinate format: '{s}'. Expected formats: 'x,y', 'x/y', 'x:y', 'WGS84 UTM 36N 735712 E / 3563829 N', 'WGS84 Geo 35° 30' 0.11\" E / 32° 11' 9.88\" N', etc."  
 except Exception as e:  
 return None, f"Error parsing coordinates '{s}': {str(e)}"  
  
def parse\_point\_simple(s):  
 """  
 Simple wrapper for backwards compatibility - returns only coordinates or None  
 """  
 result = parse\_point(s)  
 if result[0] is not None:  
 return result[0]  
 return None  
  
def calculate\_area\_size(xmin, ymin, xmax, ymax):  
 """Calculate the area size in square meters using UTM coordinates"""  
 width = abs(xmax - xmin)  
 height = abs(ymax - ymin)  
 return width \* height  
  
def calculate\_overlap\_percentage(area\_xmin, area\_ymin, area\_xmax, area\_ymax, query\_xmin, query\_ymin, query\_xmax, query\_ymax):  
 """Calculate the percentage of area that overlaps with the query rectangle"""  
 # Calculate intersection  
 intersect\_xmin = max(area\_xmin, query\_xmin)  
 intersect\_ymin = max(area\_ymin, query\_ymin)  
 intersect\_xmax = min(area\_xmax, query\_xmax)  
 intersect\_ymax = min(area\_ymax, query\_ymax)  
  
 # Check if there's an intersection  
 if intersect\_xmin >= intersect\_xmax or intersect\_ymin >= intersect\_ymax:  
 return 0.0  
  
 # Calculate areas  
 area\_size = (area\_xmax - area\_xmin) \* (area\_ymax - area\_ymin)  
 intersect\_size = (intersect\_xmax - intersect\_xmin) \* (intersect\_ymax - intersect\_ymin)  
  
 if area\_size == 0:  
 return 0.0  
  
 return (intersect\_size / area\_size) \* 100.0  
  
def convert\_date\_to\_db\_format(date\_str):  
 """Convert DD/MM/YYYY format to database format (DD-MM-YY) for comparison"""  
 try:  
 if date\_str and '/' in date\_str: # DD/MM/YYYY format  
 day, month, year = date\_str.split('/')  
 # Convert to DD-MM-YY format for database comparison  
 return f"{day.zfill(2)}-{month.zfill(2)}-{year[2:]}"  
 elif date\_str and '-' in date\_str: # DD-MM-YY format (already correct)  
 return date\_str  
 return None  
 except:  
 return None

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\utils\\_\_init\_\_.py

# Utility functions package

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\app.js

const express = require('express');  
const cors = require('cors');  
const path = require('path');  
const projectsRouter = require('./api/projects');  
const areasRouter = require('./api/areas');  
const filesRouter = require('./api/files');  
  
const app = express();  
  
// Middleware  
app.use(cors({ origin: '\*' }));  
app.use(express.json());  
app.use(express.urlencoded({ extended: true }));  
  
// Serve static files from frontend directory BEFORE API routes  
app.use(express.static(path.join(\_\_dirname, '..', 'frontend')));  
  
// API Routes  
app.use('/api', projectsRouter);  
app.use('/api', areasRouter);  
app.use('/view\_file', filesRouter);  
  
// Serve frontend HTML at root (after API routes so it doesn't interfere)  
app.get('/', (req, res) => {  
 res.sendFile(path.join(\_\_dirname, '..', 'frontend', 'index.html'));  
});  
  
// Health check endpoint  
app.get('/api/health', (req, res) => {  
 res.json({   
 status: 'healthy',   
 message: 'Backend API is running'   
 });  
});  
  
// Error handlers  
app.use((req, res, next) => {  
 res.status(404).json({ error: 'Endpoint not found' });  
});  
  
app.use((err, req, res, next) => {  
 console.error(err.stack);  
 res.status(500).json({ error: 'Internal server error' });  
});  
  
// Start server  
const PORT = process.env.PORT || 5000;  
const HOST = process.env.HOST || '0.0.0.0';  
  
app.listen(PORT, HOST, () => {  
 console.log(`Server running on http://${HOST}:${PORT}`);  
});  
  
module.exports = app;

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\install\_dependencies.bat

@echo off  
echo Installing ArcSpatialDB Node.js Backend Dependencies...  
cd /d "%~dp0"  
  
echo.  
echo Checking if Node.js is installed...  
node --version >nul 2>&1  
if %errorlevel% neq 0 (  
 echo ERROR: Node.js is not installed or not in PATH.  
 echo Please install Node.js from https://nodejs.org/  
 pause  
 exit /b 1  
)  
  
echo Node.js is installed.  
echo.  
  
echo Installing npm packages...  
call npm install  
  
if %errorlevel% equ 0 (  
 echo.  
 echo ✓ Dependencies installed successfully!  
 echo.  
 echo You can now start the server with:  
 echo npm start (production mode)  
 echo npm run dev (development mode)  
 echo start\_backend\_node.bat  
 echo.  
) else (  
 echo.  
 echo ✗ Error installing dependencies.  
 echo Please check the error messages above.  
 echo.  
)  
  
pause

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\start\_server.bat

@echo off  
echo Starting Node.js backend server on port 5000...  
cd /d "%~dp0"  
node app.js

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\test\_api.js

const http = require('http');  
  
const testEndpoints = [  
 { path: '/api/health', method: 'GET' },  
 { path: '/api/projects', method: 'GET' },  
 { path: '/api/areas', method: 'GET' }  
];  
  
function testAPI() {  
 console.log('Testing ArcSpatialDB Node.js API...\n');  
   
 testEndpoints.forEach((endpoint, index) => {  
 setTimeout(() => {  
 const options = {  
 hostname: 'localhost',  
 port: 5000,  
 path: endpoint.path,  
 method: endpoint.method  
 };  
  
 const req = http.request(options, (res) => {  
 let data = '';  
   
 res.on('data', (chunk) => {  
 data += chunk;  
 });  
   
 res.on('end', () => {  
 console.log(`${endpoint.method} ${endpoint.path}`);  
 console.log(`Status: ${res.statusCode}`);  
   
 try {  
 const json = JSON.parse(data);  
 if (endpoint.path === '/api/health') {  
 console.log(`Response: ${json.message}`);  
 } else if (endpoint.path === '/api/projects') {  
 console.log(`Projects found: ${json.projects ? json.projects.length : 0}`);  
 } else if (endpoint.path === '/api/areas') {  
 console.log(`Areas found: ${json.areas ? json.areas.length : 0}`);  
 }  
 } catch (e) {  
 console.log('Response:', data.substring(0, 100) + '...');  
 }  
   
 console.log('---\n');  
 });  
 });  
  
 req.on('error', (e) => {  
 console.error(`Error testing ${endpoint.path}:`, e.message);  
 console.log('---\n');  
 });  
  
 req.end();  
 }, index \* 1000);  
 });  
}  
  
// Wait a bit for server to start, then test  
setTimeout(testAPI, 2000);

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\test\_real\_filtering.js

const http = require('http');  
  
function testFiltering() {  
 console.log('Testing filtering with a real HTTP request...\n');  
   
 // First, test without filters  
 console.log('1. Testing without filters:');  
 const req1 = http.request({  
 hostname: 'localhost',  
 port: 5000,  
 path: '/api/projects?page=1&per\_page=3',  
 method: 'GET'  
 }, (res1) => {  
 let data1 = '';  
 res1.on('data', chunk => data1 += chunk);  
 res1.on('end', () => {  
 try {  
 const json1 = JSON.parse(data1);  
 console.log(` Found ${json1.projects.length} projects total`);  
 json1.projects.forEach(p => console.log(` - ${p.uuid}: ${p.project\_name}`));  
   
 // Now test with UUID filter using first project's UUID prefix  
 const testUuid = json1.projects[0].uuid.substring(0, 4);  
 console.log(`\n2. Testing with UUID filter "${testUuid}":`);  
   
 const req2 = http.request({  
 hostname: 'localhost',  
 port: 5000,  
 path: `/api/projects?page=1&per\_page=10&uuid\_filter=${testUuid}`,  
 method: 'GET'  
 }, (res2) => {  
 let data2 = '';  
 res2.on('data', chunk => data2 += chunk);  
 res2.on('end', () => {  
 try {  
 const json2 = JSON.parse(data2);  
 console.log(` Found ${json2.projects.length} filtered projects`);  
 json2.projects.forEach(p => console.log(` - ${p.uuid}: ${p.project\_name}`));  
   
 if (json2.projects.length < json1.projects.length) {  
 console.log('\n✅ Filtering is WORKING correctly!');  
 } else {  
 console.log('\n❌ Filtering is NOT working - same number of results');  
 }  
   
 } catch (e) {  
 console.log(' Error parsing filtered response:', e.message);  
 }  
 process.exit(0);  
 });  
 });  
   
 req2.on('error', (e) => {  
 console.log(' Error in filtered request:', e.message);  
 process.exit(1);  
 });  
   
 req2.end();  
   
 } catch (e) {  
 console.log(' Error parsing response:', e.message);  
 process.exit(1);  
 }  
 });  
 });  
   
 req1.on('error', (e) => {  
 console.log('Error in request:', e.message);  
 process.exit(1);  
 });  
   
 req1.end();  
}  
  
testFiltering();

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\api\areas.js

const express = require('express');  
const database = require('../models/database');  
const { getProjectFiles } = require('../utils/fileUtils');  
  
const router = express.Router();  
  
router.get('/areas', async (req, res) => {  
 try {  
 const page = parseInt(req.query.page) || 1;  
 const perPage = parseInt(req.query.per\_page) || 10;  
   
 // Filters  
 const filters = {  
 idFilter: req.query.id\_filter || '',  
 projectIdFilter: req.query.project\_id\_filter || '',  
 xminFilter: req.query.xmin\_filter || '',  
 yminFilter: req.query.ymin\_filter || '',  
 xmaxFilter: req.query.xmax\_filter || '',  
 ymaxFilter: req.query.ymax\_filter || '',  
 scaleFilter: req.query.scale\_filter || ''  
 };  
  
 let whereConditions = [];  
 let params = [];  
  
 // Build WHERE conditions  
 if (filters.idFilter) {  
 const idVal = parseInt(filters.idFilter);  
 if (!isNaN(idVal)) {  
 whereConditions.push('a.id = ?');  
 params.push(idVal);  
 } else {  
 whereConditions.push('a.id = ?');  
 params.push(-1); // Invalid condition to return no results  
 }  
 }  
 if (filters.projectIdFilter) {  
 whereConditions.push('a.project\_id LIKE ?');  
 params.push(`%${filters.projectIdFilter}%`);  
 }  
 if (filters.xminFilter) {  
 const xminVal = parseFloat(filters.xminFilter);  
 if (!isNaN(xminVal)) {  
 whereConditions.push('a.xmin = ?');  
 params.push(xminVal);  
 } else {  
 whereConditions.push('a.xmin = ?');  
 params.push(-1); // Invalid condition  
 }  
 }  
 if (filters.yminFilter) {  
 const yminVal = parseFloat(filters.yminFilter);  
 if (!isNaN(yminVal)) {  
 whereConditions.push('a.ymin = ?');  
 params.push(yminVal);  
 } else {  
 whereConditions.push('a.ymin = ?');  
 params.push(-1); // Invalid condition  
 }  
 }  
 if (filters.xmaxFilter) {  
 const xmaxVal = parseFloat(filters.xmaxFilter);  
 if (!isNaN(xmaxVal)) {  
 whereConditions.push('a.xmax = ?');  
 params.push(xmaxVal);  
 } else {  
 whereConditions.push('a.xmax = ?');  
 params.push(-1); // Invalid condition  
 }  
 }  
 if (filters.ymaxFilter) {  
 const ymaxVal = parseFloat(filters.ymaxFilter);  
 if (!isNaN(ymaxVal)) {  
 whereConditions.push('a.ymax = ?');  
 params.push(ymaxVal);  
 } else {  
 whereConditions.push('a.ymax = ?');  
 params.push(-1); // Invalid condition  
 }  
 }  
 if (filters.scaleFilter) {  
 whereConditions.push('a.scale LIKE ?');  
 params.push(`%${filters.scaleFilter}%`);  
 }  
  
 const whereClause = whereConditions.length > 0 ? `WHERE ${whereConditions.join(' AND ')}` : '';  
   
 // Get total count for pagination  
 const countQuery = `SELECT COUNT(\*) as total FROM areas a ${whereClause}`;  
 const countResult = await database.get(countQuery, params);  
 const totalItems = countResult.total;  
 const totalPages = Math.ceil(totalItems / perPage);  
  
 if (page > totalPages && totalPages > 0) {  
 return res.status(400).json({  
 error: 'Page number exceeds total pages',  
 total\_pages: totalPages,  
 current\_page: page  
 });  
 }  
  
 // Main query with pagination  
 const offset = (page - 1) \* perPage;  
 const query = `  
 SELECT   
 a.\*,  
 p.project\_name,  
 p.file\_location  
 FROM areas a  
 LEFT JOIN projects p ON a.project\_id = p.uuid  
 ${whereClause}  
 ORDER BY a.id  
 LIMIT ? OFFSET ?  
 `;  
 const queryParams = [...params, perPage, offset];  
  
 const areas = await database.all(query, queryParams);  
  
 // Enhance areas with file information  
 const enhancedAreas = [];  
 for (const area of areas) {  
 try {  
 if (area.file\_location) {  
 const fileInfo = getProjectFiles(area.file\_location);  
 enhancedAreas.push({  
 ...area,  
 file\_count: fileInfo.file\_count,  
 most\_recent\_file: fileInfo.most\_recent,  
 all\_files: fileInfo.all\_files,  
 // Add frontend-expected properties  
 project\_all\_files: fileInfo.all\_files,  
 project\_file\_location: area.file\_location  
 });  
 } else {  
 enhancedAreas.push({  
 ...area,  
 file\_count: 0,  
 most\_recent\_file: null,  
 all\_files: [],  
 // Add frontend-expected properties  
 project\_all\_files: [],  
 project\_file\_location: area.file\_location || ''  
 });  
 }  
 } catch (error) {  
 console.warn(`Error getting file info for area ${area.id}:`, error.message);  
 enhancedAreas.push({  
 ...area,  
 file\_count: 0,  
 most\_recent\_file: null,  
 all\_files: [],  
 // Add frontend-expected properties  
 project\_all\_files: [],  
 project\_file\_location: area.file\_location || ''  
 });  
 }  
 }  
  
 res.json({  
 areas: enhancedAreas,  
 pagination: {  
 page: page,  
 per\_page: perPage,  
 total\_items: totalItems,  
 total\_pages: totalPages,  
 has\_prev: page > 1,  
 has\_next: page < totalPages  
 },  
 filters: filters  
 });  
  
 } catch (error) {  
 console.error('Error in get\_all\_areas:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
router.get('/areas/:id', async (req, res) => {  
 try {  
 const id = parseInt(req.params.id);  
   
 if (isNaN(id)) {  
 return res.status(400).json({ error: 'Invalid area ID' });  
 }  
  
 const query = `  
 SELECT   
 a.\*,  
 p.project\_name,  
 p.file\_location  
 FROM areas a  
 LEFT JOIN projects p ON a.project\_id = p.uuid  
 WHERE a.id = ?  
 `;  
  
 const area = await database.get(query, [id]);  
   
 if (!area) {  
 return res.status(404).json({ error: 'Area not found' });  
 }  
  
 // Get file information  
 try {  
 if (area.file\_location) {  
 const fileInfo = getProjectFiles(area.file\_location);  
 area.file\_count = fileInfo.file\_count;  
 area.most\_recent\_file = fileInfo.most\_recent;  
 area.all\_files = fileInfo.all\_files;  
 // Add frontend-expected properties  
 area.project\_all\_files = fileInfo.all\_files;  
 area.project\_file\_location = area.file\_location;  
 } else {  
 area.file\_count = 0;  
 area.most\_recent\_file = null;  
 area.all\_files = [];  
 // Add frontend-expected properties  
 area.project\_all\_files = [];  
 area.project\_file\_location = '';  
 }  
 } catch (error) {  
 console.warn(`Error getting file info for area ${id}:`, error.message);  
 area.file\_count = 0;  
 area.most\_recent\_file = null;  
 area.all\_files = [];  
 // Add frontend-expected properties  
 area.project\_all\_files = [];  
 area.project\_file\_location = area.file\_location || '';  
 }  
  
 res.json(area);  
  
 } catch (error) {  
 console.error('Error in get\_area:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
module.exports = router;

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\api\files.js

const express = require('express');  
const path = require('path');  
const fs = require('fs');  
  
const router = express.Router();  
  
const PROJECT\_ROOT = path.join(\_\_dirname, '..', '..');  
  
router.get('/\*', (req, res) => {  
 try {  
 // Get the relative path from the URL  
 const relPath = req.params[0];  
 const absPath = path.resolve(path.join(PROJECT\_ROOT, relPath));  
   
 // Security: Only allow files inside project directory  
 if (!absPath.startsWith(PROJECT\_ROOT)) {  
 return res.status(403).json({ error: 'Access denied' });  
 }  
   
 if (!fs.existsSync(absPath)) {  
 return res.status(404).json({ error: 'File not found' });  
 }  
   
 // Send the file  
 res.sendFile(absPath);  
   
 } catch (error) {  
 console.error('Error serving file:', error);  
 res.status(500).json({ error: error.message });  
 }  
});  
  
module.exports = router;

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\api\projects.js

const express = require('express');  
const database = require('../models/database');  
const { parsePoint, calculateAreaSize, convertDateToDbFormat } = require('../utils/helpers');  
const { getProjectFiles } = require('../utils/fileUtils');  
  
const router = express.Router();  
  
router.get('/projects', async (req, res) => {  
 try {  
 const page = parseInt(req.query.page) || 1;  
 const perPage = parseInt(req.query.per\_page) || 10;  
   
 // Filters  
 const filters = {  
 uuidFilter: req.query.uuid\_filter || '',  
 projectNameFilter: req.query.project\_name\_filter || '',  
 userNameFilter: req.query.user\_name\_filter || '',  
 dateFilter: req.query.date\_filter || '',  
 dateFromFilter: req.query.date\_from\_filter || '',  
 dateToFilter: req.query.date\_to\_filter || '',  
 fileLocationFilter: req.query.file\_location\_filter || '',  
 paperSizeFilter: req.query.paper\_size\_filter || '',  
 associatedScalesFilter: req.query.associated\_scales\_filter || ''  
 };  
  
 let whereConditions = [];  
 let params = [];  
  
 // Build WHERE conditions  
 if (filters.uuidFilter) {  
 whereConditions.push('p.uuid LIKE ?');  
 params.push(`${filters.uuidFilter}%`);  
 }  
 if (filters.projectNameFilter) {  
 whereConditions.push('p.project\_name LIKE ?');  
 params.push(`${filters.projectNameFilter}%`);  
 }  
 if (filters.userNameFilter) {  
 whereConditions.push('p.user\_name LIKE ?');  
 params.push(`${filters.userNameFilter}%`);  
 }  
 if (filters.dateFilter) {  
 whereConditions.push('p.date LIKE ?');  
 params.push(`${filters.dateFilter}%`);  
 }  
 if (filters.fileLocationFilter) {  
 whereConditions.push('p.file\_location LIKE ?');  
 params.push(`${filters.fileLocationFilter}%`);  
 }  
 if (filters.paperSizeFilter) {  
 whereConditions.push('p.paper\_size LIKE ?');  
 params.push(`${filters.paperSizeFilter}%`);  
 }  
  
 const whereClause = whereConditions.length > 0 ? `WHERE ${whereConditions.join(' AND ')}` : '';  
   
 // Base query with aggregated scales  
 let baseQuery = `  
 SELECT   
 p.uuid,  
 p.project\_name,  
 p.user\_name,  
 p.date,  
 p.file\_location,  
 p.paper\_size,  
 p.description,  
 GROUP\_CONCAT(DISTINCT a.scale) as associated\_scales  
 FROM projects p  
 LEFT JOIN areas a ON p.uuid = a.project\_id  
 ${whereClause}  
 GROUP BY p.uuid, p.project\_name, p.user\_name, p.date, p.file\_location, p.paper\_size, p.description  
 `;  
  
 // Handle associated scales filter  
 if (filters.associatedScalesFilter) {  
 baseQuery += ` HAVING associated\_scales LIKE ?`;  
 params.push(`%${filters.associatedScalesFilter}%`);  
 }  
  
 // Get total count for pagination  
 const countQuery = `SELECT COUNT(\*) as total FROM (${baseQuery}) as subquery`;  
 const countResult = await database.get(countQuery, params);  
 const totalItems = countResult.total;  
 const totalPages = Math.ceil(totalItems / perPage);  
  
 if (page > totalPages && totalPages > 0) {  
 return res.status(400).json({  
 error: 'Page number exceeds total pages',  
 total\_pages: totalPages,  
 current\_page: page  
 });  
 }  
  
 // Add pagination  
 const offset = (page - 1) \* perPage;  
 const finalQuery = `${baseQuery} ORDER BY p.project\_name LIMIT ? OFFSET ?`;  
 const finalParams = [...params, perPage, offset];  
  
 const projects = await database.all(finalQuery, finalParams);  
  
 // Enhance projects with file information  
 const enhancedProjects = [];  
 for (const project of projects) {  
 try {  
 const fileInfo = getProjectFiles(project.file\_location);  
 const enhancedProject = {  
 ...project,  
 file\_count: fileInfo.file\_count,  
 most\_recent\_file: fileInfo.most\_recent,  
 all\_files: fileInfo.all\_files  
 };  
   
 // Add view file properties for frontend compatibility  
 if (fileInfo.most\_recent) {  
 enhancedProject.view\_file\_path = fileInfo.most\_recent.rel\_path;  
 enhancedProject.view\_file\_type = fileInfo.most\_recent.type;  
 } else {  
 enhancedProject.view\_file\_path = null;  
 enhancedProject.view\_file\_type = null;  
 }  
   
 enhancedProjects.push(enhancedProject);  
 } catch (error) {  
 console.warn(`Error getting file info for project ${project.uuid}:`, error.message);  
 enhancedProjects.push({  
 ...project,  
 file\_count: 0,  
 most\_recent\_file: null,  
 all\_files: [],  
 view\_file\_path: null,  
 view\_file\_type: null  
 });  
 }  
 }  
  
 res.json({  
 projects: enhancedProjects,  
 pagination: {  
 page: page,  
 per\_page: perPage,  
 total\_items: totalItems,  
 total\_pages: totalPages,  
 has\_prev: page > 1,  
 has\_next: page < totalPages  
 },  
 filters: filters  
 });  
  
 } catch (error) {  
 console.error('Error in get\_all\_projects:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
router.get('/projects/:uuid', async (req, res) => {  
 try {  
 const uuid = req.params.uuid;  
   
 const query = `  
 SELECT   
 p.uuid,  
 p.project\_name,  
 p.user\_name,  
 p.date,  
 p.file\_location,  
 p.paper\_size,  
 p.description,  
 GROUP\_CONCAT(DISTINCT a.scale) as associated\_scales  
 FROM projects p  
 LEFT JOIN areas a ON p.uuid = a.project\_id  
 WHERE p.uuid = ?  
 GROUP BY p.uuid, p.project\_name, p.user\_name, p.date, p.file\_location, p.paper\_size, p.description  
 `;  
  
 const project = await database.get(query, [uuid]);  
   
 if (!project) {  
 return res.status(404).json({ error: 'Project not found' });  
 }  
  
 // Get file information  
 try {  
 const fileInfo = getProjectFiles(project.file\_location);  
 project.file\_count = fileInfo.file\_count;  
 project.most\_recent\_file = fileInfo.most\_recent;  
 project.all\_files = fileInfo.all\_files;  
   
 // Add view file properties for frontend compatibility  
 if (fileInfo.most\_recent) {  
 project.view\_file\_path = fileInfo.most\_recent.rel\_path;  
 project.view\_file\_type = fileInfo.most\_recent.type;  
 } else {  
 project.view\_file\_path = null;  
 project.view\_file\_type = null;  
 }  
 } catch (error) {  
 console.warn(`Error getting file info for project ${uuid}:`, error.message);  
 project.file\_count = 0;  
 project.most\_recent\_file = null;  
 project.all\_files = [];  
 project.view\_file\_path = null;  
 project.view\_file\_type = null;  
 }  
  
 res.json(project);  
  
 } catch (error) {  
 console.error('Error in get\_project:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
router.get('/projects/:uuid/areas', async (req, res) => {  
 try {  
 const uuid = req.params.uuid;  
   
 const query = 'SELECT \* FROM areas WHERE project\_id = ? ORDER BY id';  
 const areas = await database.all(query, [uuid]);  
  
 res.json({ areas: areas });  
  
 } catch (error) {  
 console.error('Error in get\_project\_areas:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
router.post('/projects/search', async (req, res) => {  
 try {  
 const searchData = req.body;  
   
 let whereConditions = [];  
 let params = [];  
 let joinAreas = false;  
  
 // Parse spatial box  
 const bottomLeft = (searchData.bottom\_left || '').trim();  
 const topRight = (searchData.top\_right || '').trim();  
  
 if (bottomLeft && topRight) {  
 const blResult = parsePoint(bottomLeft);  
 const trResult = parsePoint(topRight);  
   
 // Check for parsing errors  
 if (blResult[1] !== null) { // Error in bottom\_left  
 return res.status(400).json({ error: `Bottom Left: ${blResult[1]}` });  
 } else if (trResult[1] !== null) { // Error in top\_right  
 return res.status(400).json({ error: `Top Right: ${trResult[1]}` });  
 } else if (!blResult[0] || !trResult[0]) { // No coordinates returned  
 return res.status(400).json({ error: 'Invalid input format. Please use X/Y or X,Y for both points.' });  
 } else {  
 const [xmin, ymin] = blResult[0];  
 const [xmax, ymax] = trResult[0];  
 if (xmin >= xmax || ymin >= ymax) {  
 return res.status(400).json({ error: 'Bottom Left must be southwest (smaller X and Y) of Top Right. Please check your input.' });  
 }  
   
 joinAreas = true;  
 // Default INSIDE spatial filter  
 whereConditions.push('a.xmin >= ? AND a.xmax <= ? AND a.ymin >= ? AND a.ymax <= ?');  
 params.push(xmin, xmax, ymin, ymax);  
 }  
 }  
  
 // Handle other search criteria  
 if (searchData.uuid && searchData.uuid.trim()) {  
 whereConditions.push('p.uuid LIKE ?');  
 params.push(`${searchData.uuid.trim()}%`);  
 }  
  
 // Handle user names search  
 if (searchData.user\_names && searchData.user\_names.length > 0) {  
 const userConditions = searchData.user\_names.map(() => 'p.user\_name LIKE ?');  
 whereConditions.push(`(${userConditions.join(' OR ')})`);  
 searchData.user\_names.forEach(name => params.push(`${name}%`));  
 }  
  
 // Handle paper size  
 if (searchData.paper\_size && searchData.paper\_size.trim()) {  
 const paperSize = searchData.paper\_size.trim();  
 if (paperSize === 'custom' && searchData.custom\_height && searchData.custom\_width) {  
 try {  
 const heightCm = parseFloat(searchData.custom\_height);  
 const widthCm = parseFloat(searchData.custom\_width);  
 const customSizeFormat = `Custom Size: Height: ${heightCm} cm, Width: ${widthCm} cm`;  
 whereConditions.push('p.paper\_size LIKE ?');  
 params.push(`${customSizeFormat}%`);  
 } catch (e) {  
 return res.status(400).json({ error: 'Custom height and width must be valid numbers.' });  
 }  
 } else if (paperSize !== 'custom') {  
 whereConditions.push('p.paper\_size LIKE ?');  
 params.push(`${paperSize}%`);  
 } else if (paperSize === 'custom' && (!searchData.custom\_height || !searchData.custom\_width)) {  
 return res.status(400).json({ error: 'Please enter both height and width for custom size.' });  
 }  
 }  
  
 // Handle scale filter  
 if (searchData.scale && searchData.scale.trim()) {  
 try {  
 const scaleVal = parseFloat(searchData.scale.trim());  
 joinAreas = true;  
 whereConditions.push('a.scale = ?');  
 params.push(scaleVal);  
 } catch (e) {  
 return res.status(400).json({ error: 'Scale must be a number.' });  
 }  
 }  
  
 // Handle date range  
 if (searchData.date\_from && searchData.date\_from.trim()) {  
 const convertedFrom = convertDateToDbFormat(searchData.date\_from.trim());  
 if (convertedFrom) {  
 whereConditions.push('p.date >= ?');  
 params.push(convertedFrom);  
 } else {  
 return res.status(400).json({ error: 'Invalid date format for "From Date". Use DD/MM/YYYY format.' });  
 }  
 }  
  
 if (searchData.date\_to && searchData.date\_to.trim()) {  
 const convertedTo = convertDateToDbFormat(searchData.date\_to.trim());  
 if (convertedTo) {  
 whereConditions.push('p.date <= ?');  
 params.push(convertedTo);  
 } else {  
 return res.status(400).json({ error: 'Invalid date format for "To Date". Use DD/MM/YYYY format.' });  
 }  
 }  
  
 // Build the query  
 const joinClause = joinAreas ? 'INNER JOIN areas a ON p.uuid = a.project\_id' : 'LEFT JOIN areas a ON p.uuid = a.project\_id';  
 const whereClause = whereConditions.length > 0 ? `WHERE ${whereConditions.join(' AND ')}` : '';  
   
 const query = `  
 SELECT   
 p.uuid,  
 p.project\_name,  
 p.user\_name,  
 p.date,  
 p.file\_location,  
 p.paper\_size,  
 p.description,  
 GROUP\_CONCAT(DISTINCT a.scale) as associated\_scales  
 FROM projects p  
 ${joinClause}  
 ${whereClause}  
 GROUP BY p.uuid, p.project\_name, p.user\_name, p.date, p.file\_location, p.paper\_size, p.description  
 ORDER BY p.project\_name  
 `;  
   
 const projects = await database.all(query, params);  
   
 // Enhance projects with file information  
 const enhancedProjects = [];  
 for (const project of projects) {  
 try {  
 const fileInfo = getProjectFiles(project.file\_location);  
 const enhancedProject = {  
 ...project,  
 file\_count: fileInfo.file\_count,  
 most\_recent\_file: fileInfo.most\_recent,  
 all\_files: fileInfo.all\_files  
 };  
   
 if (fileInfo.most\_recent) {  
 enhancedProject.view\_file\_path = fileInfo.most\_recent.rel\_path;  
 enhancedProject.view\_file\_type = fileInfo.most\_recent.type;  
 } else {  
 enhancedProject.view\_file\_path = null;  
 enhancedProject.view\_file\_type = null;  
 }  
   
 enhancedProjects.push(enhancedProject);  
 } catch (error) {  
 console.warn(`Error getting file info for project ${project.uuid}:`, error.message);  
 enhancedProjects.push({  
 ...project,  
 file\_count: 0,  
 most\_recent\_file: null,  
 all\_files: [],  
 view\_file\_path: null,  
 view\_file\_type: null  
 });  
 }  
 }  
   
 res.json({ results: enhancedProjects });  
  
 } catch (error) {  
 console.error('Error in search\_projects:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
router.delete('/projects/:uuid', async (req, res) => {  
 try {  
 const uuid = req.params.uuid;  
   
 // First delete associated areas  
 await database.run('DELETE FROM areas WHERE project\_id = ?', [uuid]);  
   
 // Then delete the project  
 const result = await database.run('DELETE FROM projects WHERE uuid = ?', [uuid]);  
   
 if (result.changes === 0) {  
 return res.status(404).json({ error: 'Project not found' });  
 }  
   
 res.json({ message: 'Project deleted successfully' });  
  
 } catch (error) {  
 console.error('Error in delete\_project:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
router.get('/projects/:uuid/files', async (req, res) => {  
 try {  
 const uuid = req.params.uuid;  
   
 // First get the project to find its file location  
 const project = await database.get('SELECT file\_location FROM projects WHERE uuid = ?', [uuid]);  
   
 if (!project) {  
 return res.status(404).json({ error: 'Project not found' });  
 }  
   
 // Get file information  
 try {  
 const fileInfo = getProjectFiles(project.file\_location);  
 res.json({  
 files: fileInfo.all\_files,  
 file\_count: fileInfo.file\_count,  
 most\_recent\_file: fileInfo.most\_recent  
 });  
 } catch (error) {  
 console.warn(`Error getting files for project ${uuid}:`, error.message);  
 res.json({  
 files: [],  
 file\_count: 0,  
 most\_recent\_file: null  
 });  
 }  
  
 } catch (error) {  
 console.error('Error in get\_project\_files:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
router.get('/user\_names', async (req, res) => {  
 try {  
 const query = 'SELECT DISTINCT user\_name FROM projects WHERE user\_name IS NOT NULL ORDER BY user\_name';  
 const results = await database.all(query, []);  
   
 const userNames = results.map(row => row.user\_name);  
   
 res.json({ user\_names: userNames });  
  
 } catch (error) {  
 console.error('Error in get\_user\_names:', error);  
 res.status(500).json({ error: 'Internal server error' });  
 }  
});  
  
module.exports = router;

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\models\database.js

const sqlite3 = require('sqlite3').verbose();  
const path = require('path');  
  
// Database configuration  
const DB\_PATH = path.join(\_\_dirname, '..', '..', 'elements.db');  
  
class Database {  
 constructor() {  
 this.db = null;  
 }  
  
 connect() {  
 return new Promise((resolve, reject) => {  
 this.db = new sqlite3.Database(DB\_PATH, (err) => {  
 if (err) {  
 console.error('Error opening database:', err.message);  
 reject(err);  
 } else {  
 console.log('Connected to SQLite database.');  
 resolve();  
 }  
 });  
 });  
 }  
  
 close() {  
 return new Promise((resolve, reject) => {  
 if (this.db) {  
 this.db.close((err) => {  
 if (err) {  
 console.error('Error closing database:', err.message);  
 reject(err);  
 } else {  
 console.log('Database connection closed.');  
 resolve();  
 }  
 });  
 } else {  
 resolve();  
 }  
 });  
 }  
  
 run(sql, params = []) {  
 return new Promise((resolve, reject) => {  
 this.db.run(sql, params, function(err) {  
 if (err) {  
 reject(err);  
 } else {  
 resolve({ id: this.lastID, changes: this.changes });  
 }  
 });  
 });  
 }  
  
 get(sql, params = []) {  
 return new Promise((resolve, reject) => {  
 this.db.get(sql, params, (err, row) => {  
 if (err) {  
 reject(err);  
 } else {  
 resolve(row);  
 }  
 });  
 });  
 }  
  
 all(sql, params = []) {  
 return new Promise((resolve, reject) => {  
 this.db.all(sql, params, (err, rows) => {  
 if (err) {  
 reject(err);  
 } else {  
 resolve(rows);  
 }  
 });  
 });  
 }  
}  
  
// Create a single instance  
const database = new Database();  
  
// Initialize database connection  
database.connect().catch(console.error);  
  
module.exports = database;

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\utils\fileUtils.js

const fs = require('fs');  
const path = require('path');  
const glob = require('glob');  
  
const PROJECT\_ROOT = path.join(\_\_dirname, '..', '..');  
  
/\*\*  
 \* Get all files (PDF, JPEG, PNG) for a project and return file information  
 \*/  
function getProjectFiles(fileLocation) {  
 const absPath = path.resolve(fileLocation);  
 const fileTypes = [  
 { ext: 'pdf', type: 'pdf' },  
 { ext: 'jpeg', type: 'img' },  
 { ext: 'jpg', type: 'img' },  
 { ext: 'png', type: 'img' }  
 ];  
   
 const allFiles = [];  
 let mostRecent = null;  
  
 for (const { ext, type } of fileTypes) {  
 const pattern = path.join(absPath, `\*.${ext}`);  
 try {  
 const files = glob.sync(pattern);  
   
 for (const file of files) {  
 const stats = fs.statSync(file);  
 const ctime = stats.ctimeMs;  
   
 const fileInfo = {  
 path: file,  
 type: type,  
 ctime: ctime,  
 filename: path.basename(file),  
 rel\_path: path.relative(PROJECT\_ROOT, file)  
 };  
   
 allFiles.push(fileInfo);  
  
 if (!mostRecent || ctime > mostRecent.ctime) {  
 mostRecent = fileInfo;  
 }  
 }  
 } catch (error) {  
 // Continue if directory doesn't exist or other errors  
 console.warn(`Warning: Could not scan directory ${absPath} for ${ext} files:`, error.message);  
 }  
 }  
  
 // Sort files by creation time (newest first)  
 allFiles.sort((a, b) => b.ctime - a.ctime);  
   
 return {  
 all\_files: allFiles,  
 file\_count: allFiles.length,  
 most\_recent: mostRecent  
 };  
}  
  
module.exports = {  
 getProjectFiles,  
 PROJECT\_ROOT  
};

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\backend\_node\utils\helpers.js

/\*\*  
 \* Parse coordinate string with support for various separators and formats.  
 \* Supports: '/', ',', ':', ';', '|', ' ', '\t', '\\', and combinations  
 \* Also handles WGS84 format and other coordinate system prefixes  
 \* Handles complex formats like:  
 \* - WGS84 UTM 36N 735712 E / 3563829 N  
 \* - WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N  
 \*   
 \* Returns: [x, y] if successful, or [null, error\_message] if failed  
 \*/  
function parsePoint(s) {  
 try {  
 s = String(s).trim();  
   
 // Check for empty or whitespace-only input  
 if (!s) {  
 return [null, "Empty coordinate string provided"];  
 }  
   
 // Handle complex WGS84 UTM format: "WGS84 UTM 36N 735712 E / 3563829 N"  
 if (s.toUpperCase().includes('WGS84 UTM')) {  
 // Pattern: WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]  
 const utmPattern = /WGS84\s+UTM\s+(\d+[NS])\s+(\d+)\s\*[EW]\s\*\/\s\*(\d+)\s\*[NS]/i;  
 const match = s.match(utmPattern);  
 if (match) {  
 try {  
 const zone = match[1];  
 const easting = parseFloat(match[2]);  
 const northing = parseFloat(match[3]);  
 return [[easting, northing], null];  
 } catch (e) {  
 return [null, `Invalid UTM coordinates in '${s}': ${e.message}`];  
 }  
 } else {  
 return [null, "Invalid WGS84 UTM format. Expected: 'WGS84 UTM [zone][N/S] [easting] [E/W] / [northing] [N/S]'"];  
 }  
 }  
   
 // Handle complex WGS84 Geographic format: "WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N"  
 if (s.toUpperCase().includes('WGS84 GEO')) {  
 // Pattern: WGS84 Geo [deg]° [min]' [sec]" [E/W] / [deg]° [min]' [sec]" [N/S]  
 const geoPattern = /WGS84\s+GEO\s+(\d+)°\s\*(\d+)'\s\*([\d.]+)"\s\*[EW]\s\*\/\s\*(\d+)°\s\*(\d+)'\s\*([\d.]+)"\s\*[NS]/i;  
 const match = s.match(geoPattern);  
 if (match) {  
 try {  
 // Convert DMS to decimal degrees  
 let lonDeg = parseFloat(match[1]);  
 const lonMin = parseFloat(match[2]);  
 const lonSec = parseFloat(match[3]);  
 let latDeg = parseFloat(match[4]);  
 const latMin = parseFloat(match[5]);  
 const latSec = parseFloat(match[6]);  
   
 // Check if longitude is East or West  
 if (s.toUpperCase().includes('W')) {  
 lonDeg = -lonDeg;  
 }  
 if (s.toUpperCase().includes('S')) {  
 latDeg = -latDeg;  
 }  
   
 // Convert to decimal degrees  
 const lonDecimal = lonDeg + (lonMin / 60) + (lonSec / 3600);  
 const latDecimal = latDeg + (latMin / 60) + (latSec / 3600);  
   
 return [[lonDecimal, latDecimal], null];  
 } catch (e) {  
 return [null, `Invalid geographic coordinates in '${s}': ${e.message}`];  
 }  
 } else {  
 return [null, "Invalid WGS84 Geographic format. Expected: 'WGS84 Geo [deg]° [min]' [sec]\" [E/W] / [deg]° [min]' [sec]\" [N/S]'"];  
 }  
 }  
   
 // Handle simple WGS84 and other coordinate system prefixes  
 const upperS = s.toUpperCase();  
 if (upperS.startsWith('WGS') || upperS.startsWith('EPSG') || upperS.startsWith('UTM') ||   
 upperS.startsWith('GEO') || upperS.startsWith('PROJ')) {  
 // Extract coordinates after the prefix  
 // Look for common patterns like "WGS84: 123.456, 789.012" or "UTM 36N: 123456, 789012"  
 const coordMatch = s.match(/[:\s]+([-\d.,\s]+)$/);  
 if (coordMatch) {  
 s = coordMatch[1].trim();  
 } else {  
 return [null, "Invalid coordinate system format. Expected: '[SYSTEM]: [x], [y]' or '[SYSTEM] [x], [y]'"];  
 }  
 }  
   
 // Remove any parentheses, brackets, or quotes  
 s = s.replace(/^[\(\)\[\]{}\"']+|[\(\)\[\]{}\"']+$/g, '');  
   
 // Try multiple separators in order of preference  
 const separators = ['/', ',', ':', ';', '|', '\\', '\t'];  
   
 // First try exact separators  
 for (const sep of separators) {  
 if (s.includes(sep)) {  
 const parts = s.split(sep, 2); // Split only on first occurrence  
 if (parts.length === 2) {  
 const xStr = parts[0].trim();  
 const yStr = parts[1].trim();  
 // Try to convert to float  
 try {  
 const x = parseFloat(xStr);  
 const y = parseFloat(yStr);  
 if (!isNaN(x) && !isNaN(y)) {  
 return [[x, y], null];  
 }  
 } catch (e) {  
 continue;  
 }  
 }  
 }  
 }  
   
 // If no separator found, try splitting on whitespace  
 if (s.includes(' ')) {  
 const parts = s.split(/\s+/);  
 if (parts.length >= 2) {  
 try {  
 const x = parseFloat(parts[0]);  
 const y = parseFloat(parts[1]);  
 if (!isNaN(x) && !isNaN(y)) {  
 return [[x, y], null];  
 }  
 } catch (e) {  
 // Continue to other patterns  
 }  
 }  
 }  
   
 // Try regex pattern for coordinates with optional spaces and various separators  
 // Pattern: number, optional spaces, separator, optional spaces, number  
 const coordPattern = /([-+]?\d\*\.?\d+)\s\*[\/,:;|\t\\]\s\*([-+]?\d\*\.?\d+)/;  
 let match = s.match(coordPattern);  
 if (match) {  
 try {  
 const x = parseFloat(match[1]);  
 const y = parseFloat(match[2]);  
 if (!isNaN(x) && !isNaN(y)) {  
 return [[x, y], null];  
 }  
 } catch (e) {  
 // Continue to other patterns  
 }  
 }  
   
 // Try pattern for coordinates separated by whitespace  
 const spacePattern = /([-+]?\d\*\.?\d+)\s+([-+]?\d\*\.?\d+)/;  
 match = s.match(spacePattern);  
 if (match) {  
 try {  
 const x = parseFloat(match[1]);  
 const y = parseFloat(match[2]);  
 if (!isNaN(x) && !isNaN(y)) {  
 return [[x, y], null];  
 }  
 } catch (e) {  
 // Continue  
 }  
 }  
   
 // If we get here, no valid format was found  
 return [null, `Invalid coordinate format: '${s}'. Expected formats: 'x,y', 'x/y', 'x:y', 'WGS84 UTM 36N 735712 E / 3563829 N', 'WGS84 Geo 35° 30' 0.11" E / 32° 11' 9.88" N', etc.`];  
 } catch (e) {  
 return [null, `Error parsing coordinates '${s}': ${e.message}`];  
 }  
}  
  
/\*\*  
 \* Simple wrapper for backwards compatibility - returns only coordinates or null  
 \*/  
function parsePointSimple(s) {  
 const result = parsePoint(s);  
 if (result[0] !== null) {  
 return result[0];  
 }  
 return null;  
}  
  
/\*\*  
 \* Calculate the area size in square meters using UTM coordinates  
 \*/  
function calculateAreaSize(xmin, ymin, xmax, ymax) {  
 const width = Math.abs(xmax - xmin);  
 const height = Math.abs(ymax - ymin);  
 return width \* height;  
}  
  
/\*\*  
 \* Calculate the percentage of area that overlaps with the query rectangle  
 \*/  
function calculateOverlapPercentage(areaXmin, areaYmin, areaXmax, areaYmax, queryXmin, queryYmin, queryXmax, queryYmax) {  
 // Calculate intersection  
 const intersectXmin = Math.max(areaXmin, queryXmin);  
 const intersectYmin = Math.max(areaYmin, queryYmin);  
 const intersectXmax = Math.min(areaXmax, queryXmax);  
 const intersectYmax = Math.min(areaYmax, queryYmax);  
  
 // Check if there's an intersection  
 if (intersectXmin >= intersectXmax || intersectYmin >= intersectYmax) {  
 return 0.0;  
 }  
  
 // Calculate areas  
 const areaSize = (areaXmax - areaXmin) \* (areaYmax - areaYmin);  
 const intersectSize = (intersectXmax - intersectXmin) \* (intersectYmax - intersectYmin);  
  
 if (areaSize === 0) {  
 return 0.0;  
 }  
  
 return (intersectSize / areaSize) \* 100.0;  
}  
  
/\*\*  
 \* Convert DD/MM/YYYY format to database format (DD-MM-YY) for comparison  
 \*/  
function convertDateToDbFormat(dateStr) {  
 try {  
 if (dateStr && dateStr.includes('/')) { // DD/MM/YYYY format  
 const [day, month, year] = dateStr.split('/');  
 // Convert to DD-MM-YY format for database comparison  
 return `${day.padStart(2, '0')}-${month.padStart(2, '0')}-${year.slice(2)}`;  
 } else if (dateStr && dateStr.includes('-')) { // DD-MM-YY format (already correct)  
 return dateStr;  
 }  
 return dateStr;  
 } catch (e) {  
 return dateStr;  
 }  
}  
  
module.exports = {  
 parsePoint,  
 parsePointSimple,  
 calculateAreaSize,  
 calculateOverlapPercentage,  
 convertDateToDbFormat  
};

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\frontend\index.html

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Project Search - ArcSpatial DB</title>  
 <link rel="stylesheet" href="css/styles.css">  
</head>  
<body>  
 <!-- Header with Logo and Copyright -->  
 <div class="header-footer">  
 <div class="logo-section">  
 <div class="logo">  
 <img src="assets/rocket.jpg" alt="Rocket Logo" style="width: 36px; height: 36px; object-fit: contain; border-radius: 50%; background: white;">  
 </div>  
 <div class="company-info">  
 <strong>ARCgis PRO DataBase</strong><br>  
 <span style="font-size: 11px;">פרוייקט שימור הידע של (נוסיף שיתאפשר)</span><br>  
 <span class="copyright">@Rocket Team Production</span>  
 </div>  
 </div>  
 <div class="copyright">  
 Version 2.0 | Spatial Database Management System  
 </div>  
 </div>  
   
 <!-- Error message container -->  
 <div id="error-message" class="error" style="display: none;"></div>  
   
 <h2>Project Search</h2>  
 <form id="searchForm">  
 <label>Bottom Left (XMin/YMin):   
 <input name="bottom\_left" type="text" placeholder="e.g., 10.5/20.1">  
 </label>  
 <label>Top Right (XMax/YMax):   
 <input name="top\_right" type="text" placeholder="e.g., 30.0/40.8">  
 </label>  
   
 <div id="relative\_size\_row" class="full-width-row">  
 <label style="display: flex; align-items: center; gap: 10px;">  
 <input name="relative\_size" id="relative\_size\_checkbox" type="checkbox" value="1"> Intersection Range  
 </label>  
 <div id="relative\_size\_percentages" style="display: none; gap: 10px; align-items: center; margin-left: 20px;">  
 <label style="margin-bottom:0;">From:   
 <input name="relative\_size\_from" type="number" min="0" max="1000" step="0.1" placeholder="e.g., 10">%  
 </label>  
 <label style="margin-bottom:0; margin-left: 10px;">To:   
 <input name="relative\_size\_to" type="number" min="0" max="1000" step="0.1" placeholder="e.g., 20">%  
 </label>  
 </div>  
 </div>  
   
 <label class="full-width-field">Project UUID:   
 <input name="uuid" type="text" placeholder="e.g., a1b2c3d4-e5f6-7890-1234-567890abcdef">  
 </label>  
   
 <div id="user\_name\_fields" style="grid-column: 1 / -1;">  
 <label>User Name:  
 <select name="user\_name">  
 <option value=""></option>  
 </select>  
 </label>  
 </div>  
 <button type="button" id="addUserNameBtn">Add another user name</button>  
   
 <div class="full-width-row" id="paper\_size\_row">  
 <label style="margin-bottom:0;">Paper Size:  
 <select name="paper\_size" id="paper\_size\_select">  
 <option value="">Select Paper Size</option>  
 <option value="A0 (Portrait)">A0 (Portrait)</option>  
 <option value="A0 (Landscape)">A0 (Landscape)</option>  
 <option value="A1 (Portrait)">A1 (Portrait)</option>  
 <option value="A1 (Landscape)">A1 (Landscape)</option>  
 <option value="A2 (Portrait)">A2 (Portrait)</option>  
 <option value="A2 (Landscape)">A2 (Landscape)</option>  
 <option value="A3 (Portrait)">A3 (Portrait)</option>  
 <option value="A3 (Landscape)">A3 (Landscape)</option>  
 <option value="A4 (Portrait)">A4 (Portrait)</option>  
 <option value="A4 (Landscape)">A4 (Landscape)</option>  
 <option value="A5 (Portrait)">A5 (Portrait)</option>  
 <option value="A5 (Landscape)">A5 (Landscape)</option>  
 <option value="B0 (Portrait)">B0 (Portrait)</option>  
 <option value="B0 (Landscape)">B0 (Landscape)</option>  
 <option value="custom">Custom Size</option>  
 </select>  
 </label>  
 <div id="custom\_size\_fields" style="display: none; margin-left: 15px; flex: 0 0 auto;">  
 <label style="margin-bottom:0;">Custom Height (cm):   
 <input name="custom\_height" type="number" step="0.1" placeholder="e.g., 29.7">  
 </label>  
 <label style="margin-bottom:0; margin-left: 10px;">Custom Width (cm):   
 <input name="custom\_width" type="number" step="0.1" placeholder="e.g., 21.0">  
 </label>  
 </div>  
 </div>  
   
 <label>Scale:   
 <input name="scale" type="text" placeholder="e.g., 1000">  
 </label>  
   
 <div id="date\_range\_fields">  
 <label>Date Range:  
 <div style="display: flex; gap: 10px; align-items: center;">  
 <input name="date\_from" type="text" placeholder="DD/MM/YYYY (e.g., 09/07/2025)" style="flex: 1;" pattern="[0-9]{2}/[0-9]{2}/[0-9]{4}">  
 <span>to</span>  
 <input name="date\_to" type="text" placeholder="DD/MM/YYYY (e.g., 25/12/2025)" style="flex: 1;" pattern="[0-9]{2}/[0-9]{2}/[0-9]{4}">  
 </div>  
 </label>  
 </div>  
   
 <div class="center-query-btn">  
 <input type="submit" value="Query">  
 <button type="button" id="resetBtn">Reset Query</button>  
 </div>  
 </form>  
  
 <!-- Search Results Section -->  
 <div id="search-results">  
 <!-- Search results will be populated here -->  
 </div>  
  
 <hr>  
  
 <!-- All Projects Section -->  
 <div id="all-projects">  
 <!-- All projects table will be populated here -->  
 </div>  
  
 <hr>  
  
 <!-- All Areas Section -->  
 <div id="all-areas">  
 <!-- All areas table will be populated here -->  
 </div>  
  
 <!-- File Modal -->  
 <div id="fileModal" class="modal">  
 <div id="fileModalContent" class="modal-content">  
 <button onclick="app.closeFileModal()" class="close">Close</button>  
 <div id="fileModalBody"></div>  
 </div>  
 </div>  
   
 <!-- Gallery Modal for All Files -->  
 <div id="galleryModal" class="gallery-modal">  
 <div id="galleryModalContent" class="gallery-modal-content">  
 <div class="gallery-header">  
 <h3 id="galleryTitle">Project Files</h3>  
 <button onclick="app.closeGalleryModal()" class="close">Close</button>  
 </div>  
   
 <div id="galleryContainer" class="gallery-container">  
 <!-- Navigation Arrows -->  
 <button id="prevBtn" onclick="app.previousFile()" class="gallery-nav-btn prev">‹</button>  
 <button id="nextBtn" onclick="app.nextFile()" class="gallery-nav-btn next">›</button>  
   
 <!-- File Display Area -->  
 <div id="galleryFileDisplay" class="gallery-file-display">  
 <!-- Content will be loaded here -->  
 </div>  
 </div>  
   
 <!-- File Info and Navigation -->  
 <div class="gallery-footer">  
 <div id="fileInfo" class="gallery-file-info">  
 <div id="fileName" class="gallery-file-name"></div>  
 <div id="fileDate" class="gallery-file-date"></div>  
 </div>  
 <div id="fileCounter" class="gallery-file-counter"></div>  
 <div id="fileType" class="gallery-file-type"></div>  
 </div>  
 </div>  
 </div>  
  
 <!-- Footer with Logo and Copyright -->  
 <div class="header-footer">  
 <div class="logo-section">  
 <div class="logo">  
 <img src="assets/rocket.jpg" alt="Rocket Logo" style="width: 36px; height: 36px; object-fit: contain; border-radius: 50%; background: white;">  
 </div>  
 <div class="company-info">  
 <strong>ARCgis PRO DataBase</strong><br>  
 <span style="font-size: 11px;">פרוייקט שימור הידע של (נוסיף שיתאפשר)</span><br>  
 <span class="copyright">@Rocket Team Production</span>  
 </div>  
 </div>  
 </div>  
  
 <script src="js/main.js"></script>  
</body>  
</html>

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\frontend\start\_frontend.bat

@echo off  
title ArcSpatialDB - Frontend Web Server  
color 0B  
echo.  
echo ========================================  
echo ArcSpatialDB Frontend Web Server  
echo ========================================  
echo.  
echo Starting frontend web server...  
echo Frontend will be available at: http://localhost:8000  
echo.  
echo Make sure the backend is running on port 5000!  
echo Press Ctrl+C to stop the server  
echo ========================================  
echo.  
  
cd /d "%~dp0"  
python -m http.server 8000  
  
echo.  
echo ========================================  
echo Server stopped. Press any key to exit.  
echo ========================================  
pause > nul

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\frontend\js\main.js

class ArcSpatialDBClient {  
 constructor(baseUrl = 'http://localhost:5000') {  
 this.baseUrl = baseUrl;  
 this.userNames = [];  
 this.currentFiles = [];  
 this.currentFileIndex = 0;  
 this.init();  
 }  
  
 async init() {  
 await this.loadUserNames();  
 this.initEventListeners();  
 this.loadAllProjects();  
 this.loadAllAreas();  
 }  
  
 // API Methods  
 async apiRequest(endpoint, options = {}) {  
 try {  
 const response = await fetch(`${this.baseUrl}${endpoint}`, {  
 headers: {  
 'Content-Type': 'application/json',  
 ...options.headers  
 },  
 ...options  
 });  
  
 if (!response.ok) {  
 const errorData = await response.json().catch(() => ({}));  
 throw new Error(errorData.error || `HTTP ${response.status}`);  
 }  
  
 return await response.json();  
 } catch (error) {  
 console.error('API Request failed:', error);  
 throw error;  
 }  
 }  
  
 async loadUserNames() {  
 try {  
 const data = await this.apiRequest('/api/user\_names');  
 this.userNames = data.user\_names;  
 this.populateUserNameDropdown();  
 } catch (error) {  
 this.showError('Failed to load user names: ' + error.message);  
 }  
 }  
  
 async searchProjects(searchData) {  
 try {  
 this.showLoading('search-results', true);  
 const data = await this.apiRequest('/api/projects/search', {  
 method: 'POST',  
 body: JSON.stringify(searchData)  
 });  
 this.displaySearchResults(data.results);  
 } catch (error) {  
 this.showError('Search failed: ' + error.message);  
 this.displaySearchResults([]);  
 } finally {  
 this.showLoading('search-results', false);  
 }  
 }  
  
 async loadAllProjects(page = 1, filters = {}) {  
 try {  
 this.showLoading('all-projects', true);  
 const params = new URLSearchParams({  
 page: page.toString(),  
 per\_page: '10',  
 ...filters  
 });  
   
 const data = await this.apiRequest(`/api/projects?${params}`);  
 this.displayAllProjects(data);  
 } catch (error) {  
 this.showError('Failed to load projects: ' + error.message);  
 } finally {  
 this.showLoading('all-projects', false);  
 }  
 }  
  
 async loadAllAreas(page = 1, filters = {}) {  
 try {  
 this.showLoading('all-areas', true);  
 const params = new URLSearchParams({  
 page: page.toString(),  
 per\_page: '10',  
 ...filters  
 });  
   
 const data = await this.apiRequest(`/api/areas?${params}`);  
 this.displayAllAreas(data);  
 } catch (error) {  
 this.showError('Failed to load areas: ' + error.message);  
 } finally {  
 this.showLoading('all-areas', false);  
 }  
 }  
  
 async deleteProject(uuid) {  
 if (!confirm('Are you sure you want to delete this project?')) {  
 return;  
 }  
  
 try {  
 await this.apiRequest(`/api/projects/${uuid}`, { method: 'DELETE' });  
 this.showSuccess('Project deleted successfully');  
 this.loadAllProjects(); // Reload projects table  
 } catch (error) {  
 this.showError('Failed to delete project: ' + error.message);  
 }  
 }  
  
 async getProjectFiles(uuid) {  
 try {  
 const data = await this.apiRequest(`/api/projects/${uuid}/files`);  
 return data;  
 } catch (error) {  
 this.showError('Failed to load project files: ' + error.message);  
 return { all\_files: [], file\_count: 0, most\_recent: null };  
 }  
 }  
  
 // UI Methods  
 initEventListeners() {  
 // Search form  
 const searchForm = document.getElementById('searchForm');  
 if (searchForm) {  
 searchForm.addEventListener('submit', (e) => {  
 e.preventDefault();  
 this.handleSearch();  
 });  
 }  
  
 // Reset button  
 const resetBtn = document.getElementById('resetBtn');  
 if (resetBtn) {  
 resetBtn.addEventListener('click', () => {  
 this.resetForm();  
 });  
 }  
  
 // Add user name button  
 const addUserNameBtn = document.getElementById('addUserNameBtn');  
 if (addUserNameBtn) {  
 addUserNameBtn.addEventListener('click', () => {  
 this.addUserNameDropdown();  
 });  
 }  
  
 // Paper size select  
 const paperSizeSelect = document.getElementById('paper\_size\_select');  
 if (paperSizeSelect) {  
 paperSizeSelect.addEventListener('change', () => {  
 this.toggleCustomSize();  
 });  
 }  
  
 // Relative size checkbox  
 const relativeSizeCheckbox = document.getElementById('relative\_size\_checkbox');  
 if (relativeSizeCheckbox) {  
 relativeSizeCheckbox.addEventListener('change', () => {  
 this.toggleRelativeSize();  
 });  
 }  
  
 // Keyboard navigation for gallery  
 document.addEventListener('keydown', (event) => {  
 if (document.getElementById('galleryModal').style.display === 'flex') {  
 if (event.key === 'ArrowLeft') {  
 this.previousFile();  
 } else if (event.key === 'ArrowRight') {  
 this.nextFile();  
 } else if (event.key === 'Escape') {  
 this.closeGalleryModal();  
 }  
 }  
 });  
 }  
  
 handleSearch() {  
 const formData = new FormData(document.getElementById('searchForm'));  
 const searchData = {};  
  
 // Convert FormData to object  
 for (let [key, value] of formData.entries()) {  
 if (key === 'user\_name') {  
 if (!searchData.user\_names) searchData.user\_names = [];  
 if (value.trim()) searchData.user\_names.push(value.trim());  
 } else if (key === 'relative\_size') {  
 searchData.relative\_size = true;  
 } else if (value.trim()) {  
 searchData[key] = value.trim();  
 }  
 }  
  
 this.searchProjects(searchData);  
 }  
  
 populateUserNameDropdown() {  
 const userNameFields = document.getElementById('user\_name\_fields');  
 const select = userNameFields.querySelector('select');  
   
 if (select) {  
 // Clear existing options except the first empty one  
 select.innerHTML = '<option value=""></option>';  
   
 // Add user names  
 this.userNames.forEach(name => {  
 const option = document.createElement('option');  
 option.value = name;  
 option.textContent = name;  
 select.appendChild(option);  
 });  
 }  
 }  
  
 addUserNameDropdown() {  
 const userNameFields = document.getElementById('user\_name\_fields');  
 const label = document.createElement('label');  
 label.innerHTML = 'User Name: ';  
   
 const select = document.createElement('select');  
 select.name = 'user\_name';  
   
 // Add empty option  
 const emptyOpt = document.createElement('option');  
 emptyOpt.value = '';  
 select.appendChild(emptyOpt);  
   
 // Add user names  
 this.userNames.forEach(name => {  
 const option = document.createElement('option');  
 option.value = name;  
 option.textContent = name;  
 select.appendChild(option);  
 });  
   
 label.appendChild(select);  
 userNameFields.appendChild(label);  
 }  
  
 toggleCustomSize() {  
 const paperSizeSelect = document.getElementById('paper\_size\_select');  
 const customFields = document.getElementById('custom\_size\_fields');  
   
 if (paperSizeSelect.value === 'custom') {  
 customFields.style.display = 'block';  
 } else {  
 customFields.style.display = 'none';  
 }  
 }  
  
 toggleRelativeSize() {  
 const checkbox = document.getElementById('relative\_size\_checkbox');  
 const percentDiv = document.getElementById('relative\_size\_percentages');  
   
 if (checkbox && percentDiv) {  
 if (checkbox.checked) {  
 percentDiv.style.display = 'flex';  
 } else {  
 percentDiv.style.display = 'none';  
 }  
 }  
 }  
  
 resetForm() {  
 document.getElementById('searchForm').reset();  
 document.getElementById('custom\_size\_fields').style.display = 'none';  
 document.getElementById('relative\_size\_percentages').style.display = 'none';  
   
 // Clear additional user name dropdowns (keep only the first one)  
 const userNamesDiv = document.getElementById('user\_name\_fields');  
 const labels = userNamesDiv.querySelectorAll('label');  
 for (let i = 1; i < labels.length; i++) {  
 labels[i].remove();  
 }  
   
 // Reset the first user name dropdown  
 if (labels.length > 0) {  
 const firstSelect = labels[0].querySelector('select');  
 if (firstSelect) {  
 firstSelect.value = '';  
 }  
 }  
  
 // Clear search results  
 this.displaySearchResults([]);  
 }  
  
 displaySearchResults(results) {  
 const container = document.getElementById('search-results');  
   
 if (!results || results.length === 0) {  
 container.innerHTML = '<p>No matching projects found.</p>';  
 return;  
 }  
  
 let html = `  
 <h3>Project Results:</h3>  
 <div class="table-container">  
 <table border="1" cellpadding="5">  
 <tr>  
 <th>UUID</th>  
 <th>Project Name</th>  
 <th>User Name</th>  
 <th>Date</th>  
 <th>File Location</th>  
 <th>Paper Size</th>  
 <th>Description</th>  
 <th>Associated Scales</th>  
 <th class="actions-column">Actions</th>  
 </tr>  
 `;  
  
 results.forEach(proj => {  
 if (proj && proj.uuid) {  
 html += `  
 <tr>  
 <td>${this.escapeHtml(proj.uuid)}</td>  
 <td>${this.escapeHtml(proj.project\_name)}</td>  
 <td>${this.escapeHtml(proj.user\_name)}</td>  
 <td>${this.escapeHtml(proj.date)}</td>  
 <td>${this.escapeHtml(proj.file\_location)}</td>  
 <td>${this.escapeHtml(proj.paper\_size)}</td>  
 <td>${this.escapeHtml(proj.description)}</td>  
 <td>${this.escapeHtml(proj.associated\_scales || 'N/A')}</td>  
 <td class="actions-column">  
 ${proj.view\_file\_path ?   
 `<a href="#" onclick="app.showFileModal('${this.baseUrl}/view\_file/${encodeURIComponent(proj.view\_file\_path)}','${proj.view\_file\_type}'); return false">View</a>` :  
 '<span>No file</span>'  
 }  
 <a href="#" onclick="app.copyPath('${this.escapeHtml(proj.file\_location)}'); return false" style="background-color: #27ae60;">Copy Path</a>  
 <button type="button" onclick="app.deleteProject('${proj.uuid}')">Delete</button>  
 </td>  
 </tr>  
 `;  
 }  
 });  
  
 html += '</table></div>';  
 container.innerHTML = html;  
 }  
  
 displayAllProjects(data) {  
 const container = document.getElementById('all-projects');  
 const projects = data.projects || [];  
 const pagination = data.pagination || {};  
  
 let html = `  
 <h2>All Projects</h2>  
 <div class="table-container">  
 <table border="1" cellpadding="5">  
 <thead>  
 <tr>  
 <th>UUID <br> <input type="text" class="filter-input" data-filter="uuid\_filter" placeholder="Filter UUID"></th>  
 <th>Project Name <br> <input type="text" class="filter-input" data-filter="project\_name\_filter" placeholder="Filter Name"></th>  
 <th>User Name <br> <input type="text" class="filter-input" data-filter="user\_name\_filter" placeholder="Filter User"></th>  
 <th>Date <br> <input type="text" class="filter-input" data-filter="date\_filter" placeholder="Filter Date"></th>  
 <th>File Location <br> <input type="text" class="filter-input" data-filter="file\_location\_filter" placeholder="Filter Location"></th>  
 <th>Paper Size <br> <input type="text" class="filter-input" data-filter="paper\_size\_filter" placeholder="Filter Size"></th>  
 <th>Description</th>  
 <th>Associated Scales <br> <input type="text" class="filter-input" data-filter="associated\_scales\_filter" placeholder="Filter Scales"></th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 `;  
  
 projects.forEach(proj => {  
 if (proj && proj.uuid) {  
 html += `  
 <tr>  
 <td>${this.escapeHtml(proj.uuid)}</td>  
 <td>${this.escapeHtml(proj.project\_name)}</td>  
 <td>${this.escapeHtml(proj.user\_name)}</td>  
 <td>${this.escapeHtml(proj.date)}</td>  
 <td>${this.escapeHtml(proj.file\_location)}</td>  
 <td>${this.escapeHtml(proj.paper\_size)}</td>  
 <td>${this.escapeHtml(proj.description)}</td>  
 <td>${this.escapeHtml(proj.associated\_scales || 'N/A')}</td>  
 <td class="actions-column">  
 ${proj.view\_file\_path ?   
 `<a href="#" onclick="app.showFileModal('${this.baseUrl}/view\_file/${encodeURIComponent(proj.view\_file\_path)}','${proj.view\_file\_type}'); return false">View</a>` :  
 '<span>No file</span>'  
 }  
 <a href="#" onclick="app.copyPath('${this.escapeHtml(proj.file\_location)}'); return false" style="background-color: #27ae60;">Copy Path</a>  
 </td>  
 </tr>  
 `;  
 }  
 });  
  
 html += `  
 </tbody>  
 </table>  
 </div>  
 `;  
  
 // Add pagination  
 if (pagination.total\_pages > 1) {  
 html += this.generatePagination(pagination, 'projects');  
 }  
  
 container.innerHTML = html;  
  
 // Add event listeners to filter inputs  
 container.querySelectorAll('.filter-input').forEach(input => {  
 input.addEventListener('change', () => {  
 this.applyProjectsTableFilters();  
 });  
 input.addEventListener('keypress', (event) => {  
 if (event.key === 'Enter') {  
 event.preventDefault();  
 this.applyProjectsTableFilters();  
 }  
 });  
 });  
 }  
  
 displayAllAreas(data) {  
 const container = document.getElementById('all-areas');  
 const areas = data.areas || [];  
 const pagination = data.pagination || {};  
  
 let html = `  
 <h2>All Areas</h2>  
 <div class="table-container">  
 <table border="1" cellpadding="5">  
 <thead>  
 <tr>  
 <th>ID <br> <input type="text" class="areas-filter-input" data-filter="id\_filter" placeholder="Filter ID"></th>  
 <th>Project UUID <br> <input type="text" class="areas-filter-input" data-filter="project\_id\_filter" placeholder="Filter UUID"></th>  
 <th>XMin <br> <input type="text" class="areas-filter-input" data-filter="xmin\_filter" placeholder="Filter XMin"></th>  
 <th>YMin <br> <input type="text" class="areas-filter-input" data-filter="ymin\_filter" placeholder="Filter YMin"></th>  
 <th>XMax <br> <input type="text" class="areas-filter-input" data-filter="xmax\_filter" placeholder="Filter XMax"></th>  
 <th>YMax <br> <input type="text" class="areas-filter-input" data-filter="ymax\_filter" placeholder="Filter YMax"></th>  
 <th>Scale <br> <input type="text" class="areas-filter-input" data-filter="scale\_filter" placeholder="Filter Scale"></th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 `;  
  
 areas.forEach(area => {  
 html += `  
 <tr>  
 <td>${area.id}</td>  
 <td>${this.escapeHtml(area.project\_id)}</td>  
 <td>${area.xmin}</td>  
 <td>${area.ymin}</td>  
 <td>${area.xmax}</td>  
 <td>${area.ymax}</td>  
 <td>${area.scale}</td>  
 <td class="actions-column">  
 <a href="#" onclick="app.showFileModalOrNoFiles(${JSON.stringify(area.project\_all\_files).replace(/"/g, '&quot;')}); return false">View Project</a>  
 <a href="#" onclick="app.copyPath('${this.escapeHtml(area.project\_file\_location)}'); return false" style="background-color: #27ae60;">Copy Project Path</a>  
 <button type="button" onclick="app.copyBottomLeft('${area.xmin}', '${area.ymin}')">Copy Bottom Left</button>  
 <button type="button" onclick="app.copyTopRight('${area.xmax}', '${area.ymax}')">Copy Top Right</button>  
 </td>  
 </tr>  
 `;  
 });  
  
 html += `  
 </tbody>  
 </table>  
 </div>  
 `;  
  
 // Add pagination  
 if (pagination.total\_pages > 1) {  
 html += this.generatePagination(pagination, 'areas');  
 }  
  
 container.innerHTML = html;  
  
 // Add event listeners to filter inputs  
 container.querySelectorAll('.areas-filter-input').forEach(input => {  
 input.addEventListener('change', () => {  
 this.applyAreasTableFilters();  
 });  
 input.addEventListener('keypress', (event) => {  
 if (event.key === 'Enter') {  
 event.preventDefault();  
 this.applyAreasTableFilters();  
 }  
 });  
 });  
 }  
  
 generatePagination(pagination, type) {  
 const { current\_page, total\_pages } = pagination;  
 let html = '<div class="pagination">';  
  
 // Previous button  
 if (current\_page > 1) {  
 html += `<a href="#" onclick="app.loadAll${type.charAt(0).toUpperCase() + type.slice(1)}(${current\_page - 1}); return false">Previous</a>`;  
 } else {  
 html += '<span class="disabled">Previous</span>';  
 }  
  
 // Page numbers  
 for (let p = 1; p <= total\_pages; p++) {  
 if (p === current\_page) {  
 html += `<span class="current-page">${p}</span>`;  
 } else {  
 html += `<a href="#" onclick="app.loadAll${type.charAt(0).toUpperCase() + type.slice(1)}(${p}); return false">${p}</a>`;  
 }  
 }  
  
 // Next button  
 if (current\_page < total\_pages) {  
 html += `<a href="#" onclick="app.loadAll${type.charAt(0).toUpperCase() + type.slice(1)}(${current\_page + 1}); return false">Next</a>`;  
 } else {  
 html += '<span class="disabled">Next</span>';  
 }  
  
 html += '</div>';  
 return html;  
 }  
  
 applyProjectsTableFilters() {  
 const filters = {};  
 document.querySelectorAll('#all-projects .filter-input').forEach(input => {  
 if (input.value.trim()) {  
 filters[input.dataset.filter] = input.value.trim();  
 }  
 });  
 this.loadAllProjects(1, filters);  
 }  
  
 applyAreasTableFilters() {  
 const filters = {};  
 document.querySelectorAll('#all-areas .areas-filter-input').forEach(input => {  
 if (input.value.trim()) {  
 filters[input.dataset.filter] = input.value.trim();  
 }  
 });  
 this.loadAllAreas(1, filters);  
 }  
  
 // Modal Methods  
 showFileModal(url, type) {  
 const modal = document.getElementById('fileModal');  
 const body = document.getElementById('fileModalBody');  
   
 if (type === 'pdf') {  
 body.innerHTML = `<iframe src="${url}" width="800" height="600" style="border:none;"></iframe>`;  
 } else if (type === 'img') {  
 body.innerHTML = `<img src="${url}" style="max-width:80vw; max-height:80vh; display:block; margin:auto;" />`;  
 }  
   
 modal.style.display = 'flex';  
 }  
  
 closeFileModal() {  
 const modal = document.getElementById('fileModal');  
 const body = document.getElementById('fileModalBody');  
 body.innerHTML = '';  
 modal.style.display = 'none';  
 }  
  
 showFileModalOrNoFiles(files) {  
 if (!files || files.length === 0) {  
 const modal = document.getElementById('fileModal');  
 const body = document.getElementById('fileModalBody');  
 body.innerHTML = '<div style="text-align:center; padding:40px; font-size:1.2em; color:#888;">No files available for this project.</div>';  
 modal.style.display = 'flex';  
 } else if (files.length === 1) {  
 const file = files[0];  
 const url = `${this.baseUrl}/view\_file/${encodeURIComponent(file.rel\_path)}`;  
 this.showFileModal(url, file.type);  
 } else {  
 // Multiple files: open gallery  
 this.showGalleryModal(files);  
 }  
 }  
  
 showGalleryModal(files) {  
 this.currentFiles = files;  
 this.currentFileIndex = 0;  
   
 const modal = document.getElementById('galleryModal');  
 const title = document.getElementById('galleryTitle');  
   
 title.textContent = `Project Files (${files.length} files)`;  
 modal.style.display = 'flex';  
   
 this.displayCurrentFile();  
 }  
  
 closeGalleryModal() {  
 const modal = document.getElementById('galleryModal');  
 modal.style.display = 'none';  
 this.currentFiles = [];  
 this.currentFileIndex = 0;  
 }  
  
 displayCurrentFile() {  
 if (this.currentFiles.length === 0) return;  
   
 const file = this.currentFiles[this.currentFileIndex];  
 const display = document.getElementById('galleryFileDisplay');  
 const fileName = document.getElementById('fileName');  
 const fileDate = document.getElementById('fileDate');  
 const fileCounter = document.getElementById('fileCounter');  
 const fileType = document.getElementById('fileType');  
 const prevBtn = document.getElementById('prevBtn');  
 const nextBtn = document.getElementById('nextBtn');  
   
 // Update file info  
 fileName.textContent = file.filename;  
 fileDate.textContent = new Date(file.ctime \* 1000).toLocaleString();  
 fileCounter.textContent = `${this.currentFileIndex + 1} / ${this.currentFiles.length}`;  
 fileType.textContent = file.type.toUpperCase();  
   
 // Update navigation buttons  
 prevBtn.style.display = this.currentFileIndex > 0 ? 'block' : 'none';  
 nextBtn.style.display = this.currentFileIndex < this.currentFiles.length - 1 ? 'block' : 'none';  
   
 // Generate URL dynamically  
 const fileUrl = `${this.baseUrl}/view\_file/${encodeURIComponent(file.rel\_path)}`;  
   
 // Display file content  
 if (file.type === 'pdf') {  
 display.innerHTML = `<iframe src="${fileUrl}" width="800" height="600" style="border:none; max-width:100%; max-height:100%;"></iframe>`;  
 } else {  
 display.innerHTML = `<img src="${fileUrl}" style="max-width:100%; max-height:100%; object-fit:contain;" alt="${file.filename}">`;  
 }  
 }  
  
 previousFile() {  
 if (this.currentFileIndex > 0) {  
 this.currentFileIndex--;  
 this.displayCurrentFile();  
 }  
 }  
  
 nextFile() {  
 if (this.currentFileIndex < this.currentFiles.length - 1) {  
 this.currentFileIndex++;  
 this.displayCurrentFile();  
 }  
 }  
  
 // Utility Methods  
 copyPath(path) {  
 navigator.clipboard.writeText(path).then(() => {  
 this.showCopyNotification('Path copied to clipboard!');  
 }).catch(() => {  
 // Fallback for older browsers  
 const textarea = document.createElement('textarea');  
 textarea.value = path;  
 textarea.style.position = 'fixed';  
 textarea.style.opacity = '0';  
 document.body.appendChild(textarea);  
 textarea.select();  
 document.execCommand('copy');  
 document.body.removeChild(textarea);  
 this.showCopyNotification('Path copied to clipboard!');  
 });  
 }  
  
 copyTopRight(xmax, ymax) {  
 const str = `${xmax}/${ymax}`;  
 this.copyToClipboard(str, `Top Right copied: ${str}`);  
 }  
  
 copyBottomLeft(xmin, ymin) {  
 const str = `${xmin}/${ymin}`;  
 this.copyToClipboard(str, `Bottom Left copied: ${str}`);  
 }  
  
 copyToClipboard(text, message) {  
 navigator.clipboard.writeText(text).then(() => {  
 this.showCopyNotification(message);  
 }).catch(() => {  
 // Fallback for older browsers  
 const textarea = document.createElement('textarea');  
 textarea.value = text;  
 textarea.style.position = 'fixed';  
 textarea.style.opacity = '0';  
 document.body.appendChild(textarea);  
 textarea.select();  
 document.execCommand('copy');  
 document.body.removeChild(textarea);  
 this.showCopyNotification(message);  
 });  
 }  
  
 showCopyNotification(message) {  
 const notification = document.createElement('div');  
 notification.textContent = message;  
 notification.style.cssText = 'position: fixed; top: 20px; right: 20px; background: #27ae60; color: white; padding: 10px 15px; border-radius: 5px; z-index: 10000; font-size: 14px;';  
 document.body.appendChild(notification);  
   
 setTimeout(() => {  
 if (notification.parentNode) {  
 notification.parentNode.removeChild(notification);  
 }  
 }, 2000);  
 }  
  
 showError(message) {  
 const errorDiv = document.getElementById('error-message');  
 if (errorDiv) {  
 errorDiv.textContent = message;  
 errorDiv.style.display = 'block';  
 setTimeout(() => {  
 errorDiv.style.display = 'none';  
 }, 5000);  
 } else {  
 alert('Error: ' + message);  
 }  
 }  
  
 showSuccess(message) {  
 this.showCopyNotification(message);  
 }  
  
 showLoading(containerId, show) {  
 const container = document.getElementById(containerId);  
 if (!container) return;  
  
 let loadingDiv = container.querySelector('.loading');  
 if (!loadingDiv) {  
 loadingDiv = document.createElement('div');  
 loadingDiv.className = 'loading';  
 loadingDiv.textContent = 'Loading...';  
 container.appendChild(loadingDiv);  
 }  
  
 loadingDiv.style.display = show ? 'block' : 'none';  
 }  
  
 escapeHtml(text) {  
 if (typeof text !== 'string') return text;  
 const div = document.createElement('div');  
 div.textContent = text;  
 return div.innerHTML;  
 }  
}  
  
// Initialize the application  
let app;  
document.addEventListener('DOMContentLoaded', () => {  
 app = new ArcSpatialDBClient();  
});

--------------------------------------------------

## File: C:\Users\yuval\PycharmProjects\ArcSpatialDB\templates\index.html

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Project Search</title>  
 <style>  
 body {  
 font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
 margin: 20px;  
 background-color: #f4f7f6;  
 color: #333;  
 line-height: 1.6;  
 }  
   
 .header-footer {  
 background: linear-gradient(135deg, #2c3e50 0%, #3498db 100%);  
 color: white;  
 padding: 15px 20px;  
 border-radius: 8px;  
 margin-bottom: 20px;  
 display: flex;  
 align-items: center;  
 justify-content: space-between;  
 box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
 }  
   
 .header-footer:last-of-type {  
 margin-top: 30px;  
 margin-bottom: 0;  
 }  
   
 .logo-section {  
 display: flex;  
 align-items: center;  
 gap: 10px;  
 }  
   
 .logo {  
 width: 40px;  
 height: 40px;  
 background: #e74c3c;  
 border-radius: 50%;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 font-weight: bold;  
 font-size: 18px;  
 color: white;  
 }  
   
 .company-info {  
 font-size: 14px;  
 }  
   
 .copyright {  
 font-size: 12px;  
 opacity: 0.9;  
 }  
  
 h2, h3 {  
 color: #2c3e50;  
 border-bottom: 2px solid #3498db;  
 padding-bottom: 10px;  
 margin-top: 30px;  
 }  
  
 form {  
 background-color: #ffffff;  
 padding: 25px;  
 border-radius: 8px;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 margin-bottom: 30px;  
 display: grid;  
 gap: 15px;  
 grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));  
 }  
   
 /\* Force specific fields to be on their own line \*/  
 .full-width-field {  
 grid-column: 1 / -1;  
 }  
  
 form label {  
 display: flex;  
 flex-direction: column;  
 font-weight: bold;  
 margin-bottom: 5px;  
 }  
  
 form input[type="text"],  
 form select {  
 padding: 10px;  
 border: 1px solid #ddd;  
 border-radius: 4px;  
 font-size: 1em;  
 width: 100%;  
 box-sizing: border-box;  
 }  
  
 form button,  
 form input[type="submit"] {  
 background-color: #3498db;  
 color: white;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
 font-size: 1em;  
 transition: background-color 0.3s ease;  
 margin-top: 10px;  
 }  
  
 form button:hover,  
 form input[type="submit"]:hover {  
 background-color: #2980b9;  
 }  
  
 #user\_name\_fields {  
 grid-column: span 2;  
 display: flex;  
 flex-wrap: wrap;  
 gap: 15px;  
 align-items: flex-end;  
 }  
  
 #user\_name\_fields label {  
 flex: 1 1 auto;  
 min-width: 200px;  
 }  
  
 .error {  
 color: red;  
 background-color: #ffe5e5;  
 border: 1px solid red;  
 padding: 10px;  
 border-radius: 5px;  
 margin-bottom: 20px;  
 }  
  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-top: 20px;  
 background-color: #ffffff;  
 box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
 border-radius: 8px;  
 overflow: hidden;  
 }  
  
 th, td {  
 padding: 12px 15px;  
 border: 1px solid #e0e0e0;  
 text-align: left;  
 }  
  
 th {  
 background-color: #3498db;  
 color: white;  
 font-weight: bold;  
 position: sticky;  
 top: 0;  
 z-index: 1;  
 }  
  
 tr:nth-child(even) {  
 background-color: #f9f9f9;  
 }  
  
 tr:hover {  
 background-color: #f1f1f1;  
 }  
  
 p {  
 margin-top: 15px;  
 font-style: italic;  
 color: #555;  
 }  
  
 hr {  
 border: 0;  
 height: 1px;  
 background-image: linear-gradient(to right, rgba(0, 0, 0, 0), rgba(0, 0, 0, 0.75), rgba(0, 0, 0, 0));  
 margin: 40px 0;  
 }  
  
 .pagination {  
 margin-top: 20px;  
 text-align: center;  
 }  
  
 .pagination a, .pagination span {  
 display: inline-block;  
 padding: 8px 16px;  
 margin: 0 4px;  
 border: 1px solid #ddd;  
 border-radius: 4px;  
 text-decoration: none;  
 color: #3498db;  
 background-color: #fff;  
 transition: background-color 0.3s, color 0.3s;  
 }  
  
 .pagination a:hover {  
 background-color: #3498db;  
 color: white;  
 }  
  
 .pagination span.current-page {  
 background-color: #3498db;  
 color: white;  
 border-color: #3498db;  
 font-weight: bold;  
 }  
  
 .pagination span.disabled {  
 color: #bbb;  
 cursor: not-allowed;  
 }  
 .filter-form input[type="text"] {  
 width: calc(100% - 10px); /\* Adjust width for padding \*/  
 padding: 5px;  
 margin: 2px 0;  
 box-sizing: border-box;  
 border: 1px solid #ccc;  
 border-radius: 3px;  
 }  
   
 .filter-form input[type="date"] {  
 padding: 3px;  
 margin: 1px 0;  
 box-sizing: border-box;  
 border: 1px solid #ccc;  
 border-radius: 3px;  
 font-size: 0.8em;  
 }  
 .table-container {  
 overflow-x: auto; /\* Enable horizontal scrolling for tables \*/  
 }  
  
 /\* Date range styling \*/  
 #date\_range\_fields {  
 grid-column: span 2;  
 }  
   
 #date\_range\_fields input[type="text"] {  
 padding: 10px;  
 border: 1px solid #ddd;  
 border-radius: 4px;  
 font-size: 1em;  
 width: 100%;  
 box-sizing: border-box;  
 }  
   
 #date\_range\_fields span {  
 font-weight: bold;  
 color: #666;  
 }  
  
 /\* Custom size fields styling \*/  
 #custom\_size\_fields {  
 grid-column: span 2;  
 display: flex;  
 gap: 15px;  
 align-items: flex-end;  
 }  
  
 #custom\_size\_fields label {  
 flex: 1 1 auto;  
 min-width: 200px;  
 }  
  
 #custom\_size\_fields input[type="number"] {  
 padding: 10px;  
 border: 1px solid #ddd;  
 border-radius: 4px;  
 font-size: 1em;  
 width: 100%;  
 box-sizing: border-box;  
 }  
  
 .full-width-row {  
 grid-column: 1 / -1;  
 width: 100%;  
 display: flex;  
 align-items: flex-end;  
 margin-top: 10px;  
 }  
  
 /\* Responsive adjustments \*/  
 @media (max-width: 768px) {  
 form {  
 grid-template-columns: 1fr;  
 }  
 #user\_name\_fields {  
 grid-column: span 1;  
 flex-direction: column;  
 }  
 #date\_range\_fields {  
 grid-column: span 1;  
 }  
 #custom\_size\_fields {  
 grid-column: span 1;  
 flex-direction: column;  
 }  
 #spatial\_options\_fields {  
 grid-column: span 1;  
 flex-direction: column;  
 gap: 10px;  
 }  
 #percentage\_inputs\_fields {  
 grid-column: span 1;  
 flex-direction: column;  
 gap: 10px;  
 }  
 }  
   
 /\* Spatial options styling \*/  
 #spatial\_options\_fields {  
 grid-column: span 2;  
 display: flex;  
 flex-wrap: wrap;  
 gap: 20px;  
 align-items: center;  
 margin-top: 10px;  
 padding: 15px;  
 background-color: #f8f9fa;  
 border-radius: 8px;  
 border: 1px solid #e9ecef;  
 }  
   
 #spatial\_options\_fields input[type="checkbox"] {  
 margin-right: 8px;  
 transform: scale(1.2);  
 }  
   
 #spatial\_options\_fields label {  
 display: flex;  
 align-items: center;  
 margin-bottom: 0;  
 font-weight: normal;  
 white-space: nowrap;  
 }  
   
 /\* Percentage inputs styling \*/  
 #percentage\_inputs\_fields {  
 grid-column: span 2;  
 display: flex;  
 gap: 20px;  
 align-items: center;  
 margin-top: 10px;  
 }  
   
 #percentage\_inputs\_fields input[type="number"] {  
 padding: 8px;  
 border: 1px solid #ddd;  
 border-radius: 4px;  
 font-size: 1em;  
 width: 120px;  
 box-sizing: border-box;  
 }  
   
 #percentage\_inputs\_fields label {  
 display: flex;  
 align-items: center;  
 margin-bottom: 0;  
 font-weight: normal;  
 }  
   
 .percentage-input {  
 display: none;  
 }  
   
 .overlap-percentage-input {  
 display: none;  
 }  
 .center-query-btn {  
 grid-column: 1 / -1;  
 width: 100%;  
 display: flex;  
 justify-content: center;  
 margin-top: 20px;  
 }  
 .center-query-btn input[type="submit"] {  
 min-width: 200px;  
 }  
   
 .center-query-btn button[type="button"] {  
 min-width: 200px;  
 background-color: #e74c3c;  
 color: white;  
 padding: 10px 20px;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
 font-size: 1em;  
 transition: background-color 0.3s ease;  
 margin-left: 10px;  
 }  
   
 .center-query-btn button[type="button"]:hover {  
 background-color: #c0392b;  
 }  
   
 /\* Actions column styling \*/  
 .actions-column {  
 min-width: 120px;  
 }  
   
 .actions-column a {  
 display: inline-block;  
 margin: 2px 0;  
 padding: 4px 8px;  
 background-color: #3498db;  
 color: white;  
 text-decoration: none;  
 border-radius: 3px;  
 font-size: 0.9em;  
 }  
   
 .actions-column a:hover {  
 background-color: #2980b9;  
 }  
  
 .actions-column form {  
 display: inline;  
 margin: 0;  
 padding: 0;  
 }  
 .actions-column button[type="submit"] {  
 display: inline-block;  
 margin: 2px 0;  
 padding: 4px 8px;  
 background-color: #e74c3c;  
 color: white;  
 text-decoration: none;  
 border-radius: 3px;  
 font-size: 0.9em;  
 border: none;  
 cursor: pointer;  
 transition: background-color 0.3s;  
 vertical-align: middle;  
 }  
 .actions-column button[type="submit"]:hover {  
 background-color: #c0392b;  
 }  
 </style>  
</head>  
<body>  
 <!-- Header with Logo and Copyright -->  
 <div class="header-footer">  
 <div class="logo-section">  
 <div class="logo">  
 <img src="{{ url\_for('static', filename='rocket.jpg') }}" alt="Rocket Logo" style="width: 36px; height: 36px; object-fit: contain; border-radius: 50%; background: white;">  
 </div>  
 <div class="company-info">  
 <strong>ARCgis PRO DataBase</strong><br>  
 <span style="font-size: 11px;">פרוייקט שימור הידע של (נוסיף שיתאפשר)</span><br>  
 <span class="copyright">@Rocket Team Production</span>  
 </div>  
 </div>  
 <div class="copyright">  
 Version 1.0 | Spatial Database Management System  
 </div>  
 </div>  
   
 <h2>Project Search</h2>  
 <form method="post" id="searchForm">  
 <label>Bottom Left (XMin/YMin): <input name="bottom\_left" type="text" placeholder="e.g., 10.5/20.1" value="{{ request.form.bottom\_left if request.form.bottom\_left else '' }}"></label>  
 <label>Top Right (XMax/YMax): <input name="top\_right" type="text" placeholder="e.g., 30.0/40.8" value="{{ request.form.top\_right if request.form.top\_right else '' }}"></label>  
 <div id="relative\_size\_row" class="full-width-row">  
 <label style="display: flex; align-items: center; gap: 10px;">  
 <input name="relative\_size" id="relative\_size\_checkbox" type="checkbox" value="1" {% if request.form.relative\_size %}checked{% endif %} onchange="toggleRelativeSize()"> Intersection Range  
 </label>  
 <div id="relative\_size\_percentages" style="display: none; gap: 10px; align-items: center; margin-left: 20px;">  
 <label style="margin-bottom:0;">From: <input name="relative\_size\_from" type="number" min="0" max="1000" step="0.1" placeholder="e.g., 10" value="{{ request.form.relative\_size\_from if request.form.relative\_size\_from else '' }}">%</label>  
 <label style="margin-bottom:0; margin-left: 10px;">To: <input name="relative\_size\_to" type="number" min="0" max="1000" step="0.1" placeholder="e.g., 20" value="{{ request.form.relative\_size\_to if request.form.relative\_size\_to else '' }}">%</label>  
 </div>  
 </div>  
 <label class="full-width-field">Project UUID: <input name="uuid" type="text" placeholder="e.g., a1b2c3d4-e5f6-7890-1234-567890abcdef" value="{{ request.form.uuid if request.form.uuid else '' }}"></label>  
 <div id="user\_name\_fields" style="grid-column: 1 / -1;">  
 <label>User Name:  
 <select name="user\_name">  
 <option value=""></option>  
 {% for name in user\_names %}  
 <option value="{{ name }}">{{ name }}</option>  
 {% endfor %}  
 </select>  
 </label>  
 </div>  
 <button type="button" onclick="addUserNameDropdown()">Add another user name</button>  
 <div class="full-width-row" id="paper\_size\_row">  
 <label style="margin-bottom:0;">Paper Size:  
 <select name="paper\_size" id="paper\_size\_select" onchange="toggleCustomSize()">  
 <option value="">Select Paper Size</option>  
 <option value="A0 (Portrait)" {% if request.form.paper\_size == 'A0 (Portrait)' %}selected{% endif %}>A0 (Portrait)</option>  
 <option value="A0 (Landscape)" {% if request.form.paper\_size == 'A0 (Landscape)' %}selected{% endif %}>A0 (Landscape)</option>  
 <option value="A1 (Portrait)" {% if request.form.paper\_size == 'A1 (Portrait)' %}selected{% endif %}>A1 (Portrait)</option>  
 <option value="A1 (Landscape)" {% if request.form.paper\_size == 'A1 (Landscape)' %}selected{% endif %}>A1 (Landscape)</option>  
 <option value="A2 (Portrait)" {% if request.form.paper\_size == 'A2 (Portrait)' %}selected{% endif %}>A2 (Portrait)</option>  
 <option value="A2 (Landscape)" {% if request.form.paper\_size == 'A2 (Landscape)' %}selected{% endif %}>A2 (Landscape)</option>  
 <option value="A3 (Portrait)" {% if request.form.paper\_size == 'A3 (Portrait)' %}selected{% endif %}>A3 (Portrait)</option>  
 <option value="A3 (Landscape)" {% if request.form.paper\_size == 'A3 (Landscape)' %}selected{% endif %}>A3 (Landscape)</option>  
 <option value="A4 (Portrait)" {% if request.form.paper\_size == 'A4 (Portrait)' %}selected{% endif %}>A4 (Portrait)</option>  
 <option value="A4 (Landscape)" {% if request.form.paper\_size == 'A4 (Landscape)' %}selected{% endif %}>A4 (Landscape)</option>  
 <option value="A5 (Portrait)" {% if request.form.paper\_size == 'A5 (Portrait)' %}selected{% endif %}>A5 (Portrait)</option>  
 <option value="A5 (Landscape)" {% if request.form.paper\_size == 'A5 (Landscape)' %}selected{% endif %}>A5 (Landscape)</option>  
 <option value="B0 (Portrait)" {% if request.form.paper\_size == 'B0 (Portrait)' %}selected{% endif %}>B0 (Portrait)</option>  
 <option value="B0 (Landscape)" {% if request.form.paper\_size == 'B0 (Landscape)' %}selected{% endif %}>B0 (Landscape)</option>  
 <option value="custom" {% if request.form.paper\_size == 'custom' %}selected{% endif %}>Custom Size</option>  
 </select>  
 </label>  
 <div id="custom\_size\_fields" style="display: none; margin-left: 15px; flex: 0 0 auto;">  
 <label style="margin-bottom:0;">Custom Height (cm): <input name="custom\_height" type="number" step="0.1" placeholder="e.g., 29.7" value="{{ request.form.custom\_height if request.form.custom\_height else '' }}"></label>  
 <label style="margin-bottom:0; margin-left: 10px;">Custom Width (cm): <input name="custom\_width" type="number" step="0.1" placeholder="e.g., 21.0" value="{{ request.form.custom\_width if request.form.custom\_width else '' }}"></label>  
 </div>  
 </div>  
 <label>Scale: <input name="scale" type="text" placeholder="e.g., 1000" value="{{ request.form.scale if request.form.scale else '' }}"></label>  
 <div id="date\_range\_fields">  
 <label>Date Range:  
 <div style="display: flex; gap: 10px; align-items: center;">  
 <input name="date\_from" type="text" placeholder="DD/MM/YYYY (e.g., 09/07/2025)" value="{{ request.form.date\_from if request.form.date\_from else '' }}" style="flex: 1;" pattern="[0-9]{2}/[0-9]{2}/[0-9]{4}">  
 <span>to</span>  
 <input name="date\_to" type="text" placeholder="DD/MM/YYYY (e.g., 25/12/2025)" value="{{ request.form.date\_to if request.form.date\_to else '' }}" style="flex: 1;" pattern="[0-9]{2}/[0-9]{2}/[0-9]{4}">  
 </div>  
 </label>  
 </div>  
 <div class="center-query-btn">  
 <input type="submit" value="Query">  
 <button type="button" onclick="resetForm()">Reset Query</button>  
 </div>  
 </form>  
 <script>  
 function addUserNameDropdown() {  
 var userNames = {{ user\_names|tojson }};  
 var div = document.getElementById('user\_name\_fields');  
 var label = document.createElement('label');  
 label.innerHTML = 'User Name: ';  
 var select = document.createElement('select');  
 select.name = 'user\_name';  
 var emptyOpt = document.createElement('option');  
 emptyOpt.value = '';  
 select.appendChild(emptyOpt);  
 for (var i = 0; i < userNames.length; i++) {  
 var opt = document.createElement('option');  
 opt.value = userNames[i];  
 opt.text = userNames[i];  
 select.appendChild(opt);  
 }  
 label.appendChild(select);  
 div.appendChild(label);  
 }  
  
 function toggleCustomSize() {  
 var paperSizeSelect = document.getElementById('paper\_size\_select');  
 var customFields = document.getElementById('custom\_size\_fields');  
   
 if (paperSizeSelect.value === 'custom') {  
 customFields.style.display = 'block';  
 } else {  
 customFields.style.display = 'none';  
 }  
 }  
  
 function toggleRelativeSize() {  
 var checkbox = document.getElementById('relative\_size\_checkbox');  
 var percentDiv = document.getElementById('relative\_size\_percentages');  
 if (checkbox && percentDiv) {  
 if (checkbox.checked) {  
 percentDiv.style.display = 'flex';  
 } else {  
 percentDiv.style.display = 'none';  
 }  
 }  
 }  
  
 function resetForm() {  
 // Reset all form inputs  
 document.getElementById('searchForm').reset();  
   
 // Hide custom size fields  
 document.getElementById('custom\_size\_fields').style.display = 'none';  
 // Hide relative size percentages  
 document.getElementById('relative\_size\_percentages').style.display = 'none';  
   
 // Clear any additional user name dropdowns (keep only the first one)  
 var userNamesDiv = document.getElementById('user\_name\_fields');  
 var labels = userNamesDiv.querySelectorAll('label');  
 for (var i = 1; i < labels.length; i++) {  
 labels[i].remove();  
 }  
   
 // Reset the first user name dropdown  
 if (labels.length > 0) {  
 var firstSelect = labels[0].querySelector('select');  
 if (firstSelect) {  
 firstSelect.value = '';  
 }  
 }  
   
 // Clear URL parameters and reload the page to reset everything  
 window.location.href = window.location.pathname;  
 }  
  
 function copyPath(path) {  
 // Create a temporary textarea element  
 var textarea = document.createElement('textarea');  
 textarea.value = path;  
 textarea.style.position = 'fixed';  
 textarea.style.opacity = '0';  
 document.body.appendChild(textarea);  
   
 // Select and copy the text  
 textarea.select();  
 document.execCommand('copy');  
   
 // Remove the temporary element  
 document.body.removeChild(textarea);  
   
 // Show a brief notification  
 var notification = document.createElement('div');  
 notification.textContent = 'Path copied to clipboard!';  
 notification.style.cssText = 'position: fixed; top: 20px; right: 20px; background: #27ae60; color: white; padding: 10px 15px; border-radius: 5px; z-index: 10000; font-size: 14px;';  
 document.body.appendChild(notification);  
   
 // Remove notification after 2 seconds  
 setTimeout(function() {  
 if (notification.parentNode) {  
 notification.parentNode.removeChild(notification);  
 }  
 }, 2000);  
 }  
  
 // Function to get a URL parameter  
 function getUrlParameter(name) {  
 name = name.replace(/[\[]/, '\\[').replace(/[\]]/, '\\]');  
 var regex = new RegExp('[\\?&]' + name + '=([^&#]\*)');  
 var results = regex.exec(location.search);  
 return results === null ? '' : decodeURIComponent(results[1].replace(/\+/g, ' '));  
 };  
 function applyProjectsTableFilters() {  
 let filters = {  
 projects\_uuid\_filter: document.querySelector('input[name="projects\_uuid\_filter"]').value,  
 projects\_project\_name\_filter: document.querySelector('input[name="projects\_project\_name\_filter"]').value,  
 projects\_user\_name\_filter: document.querySelector('input[name="projects\_user\_name\_filter"]').value,  
 projects\_date\_filter: document.querySelector('input[name="projects\_date\_filter"]').value,  
 projects\_date\_from\_filter: document.querySelector('input[name="projects\_date\_from\_filter"]').value,  
 projects\_date\_to\_filter: document.querySelector('input[name="projects\_date\_to\_filter"]').value,  
 projects\_file\_location\_filter: document.querySelector('input[name="projects\_file\_location\_filter"]').value,  
 projects\_paper\_size\_filter: document.querySelector('input[name="projects\_paper\_size\_filter"]').value,  
 projects\_associated\_scales\_filter: document.querySelector('input[name="projects\_associated\_scales\_filter"]').value // The new filter  
 };  
 let queryParams = new URLSearchParams(window.location.search);  
 for (let key in filters) {  
 if (filters[key]) {  
 queryParams.set(key, filters[key]);  
 } else {  
 queryParams.delete(key);  
 }  
 }  
 queryParams.delete('page'); // Reset page when filters change  
 window.location.search = queryParams.toString();  
 }  
 // -----------------------------------------------------  
 window.onload = function() {  
 var selected = {{ selected\_user\_names|tojson }};  
 var userNames = {{ user\_names|tojson }};  
 var div = document.getElementById('user\_name\_fields');  
  
 // Clear existing dropdowns if we are re-populating with multiple or if initial is empty and we have selections  
 if (selected.length > 1 || (selected.length === 1 && selected[0] !== '' && div.querySelector('select').value === '')) {  
 div.innerHTML = '';  
 } else if (selected.length === 1 && selected[0] !== '') {  
 // If only one selected and it's not empty, just set its value and return  
 var firstSelect = div.querySelector('select');  
 if (firstSelect) {  
 firstSelect.value = selected[0];  
 }  
 // Do not return here if you want other window.onload logic to run  
 }  
  
 for (var j = 0; j < selected.length; j++) {  
 var label = document.createElement('label');  
 label.innerHTML = 'User Name: ';  
 var select = document.createElement('select');  
 select.name = 'user\_name';  
 var emptyOpt = document.createElement('option');  
 emptyOpt.value = '';  
 select.appendChild(emptyOpt);  
 for (var i = 0; i < userNames.length; i++) {  
 var opt = document.createElement('option');  
 opt.value = userNames[i];  
 opt.text = userNames[i];  
 if (userNames[i] == selected[j]) opt.selected = true;  
 select.appendChild(opt);  
 }  
 label.appendChild(select);  
 div.appendChild(label);  
 }  
  
 // Add hidden input for scroll position to each filter form  
 const projectsFilterForm = document.getElementById('projectsFilterForm');  
 if (projectsFilterForm) {  
 let scrollInput = document.createElement('input');  
 scrollInput.type = 'hidden';  
 scrollInput.name = 'scroll\_pos';  
 scrollInput.id = 'projects\_scroll\_pos';  
 projectsFilterForm.appendChild(scrollInput);  
  
 projectsFilterForm.addEventListener('submit', function() {  
 document.getElementById('projects\_scroll\_pos').value = window.scrollY;  
 });  
 }  
  
 const areasFilterForm = document.getElementById('areasFilterForm');  
 if (areasFilterForm) {  
 let scrollInput = document.createElement('input');  
 scrollInput.type = 'hidden';  
 scrollInput.name = 'scroll\_pos';  
 scrollInput.id = 'areas\_scroll\_pos';  
 areasFilterForm.appendChild(scrollInput);  
  
 areasFilterForm.addEventListener('submit', function() {  
 document.getElementById('areas\_scroll\_pos').value = window.scrollY;  
 });  
 }  
  
 // Add event listeners to filter inputs for submitting on change (or enter key)  
 document.querySelectorAll('.filter-input').forEach(input => {  
 input.addEventListener('change', function() {  
 // Ensure the hidden scroll input is updated before submission  
 if (this.closest('form') && this.closest('form').id === 'projectsFilterForm') {  
 document.getElementById('projects\_scroll\_pos').value = window.scrollY;  
 } else if (this.closest('form') && this.closest('form').id === 'areasFilterForm') {  
 document.getElementById('areas\_scroll\_pos').value = window.scrollY;  
 }  
 this.form.submit();  
 });  
 input.addEventListener('keypress', function(event) {  
 if (event.key === 'Enter') {  
 event.preventDefault(); // Prevent default Enter key behavior (form submission)  
 // Ensure the hidden scroll input is updated before submission  
 if (this.closest('form') && this.closest('form').id === 'projectsFilterForm') {  
 document.getElementById('projects\_scroll\_pos').value = window.scrollY;  
 } else if (this.closest('form') && this.closest('form').id === 'areasFilterForm') {  
 document.getElementById('areas\_scroll\_pos').value = window.scrollY;  
 }  
 this.form.submit();  
 }  
 });  
 });  
  
 // Restore scroll position after the page loads  
 const scroll\_pos = getUrlParameter('scroll\_pos');  
 if (scroll\_pos) {  
 window.scrollTo(0, parseInt(scroll\_pos));  
 }  
  
 // Initialize custom size fields state  
 toggleCustomSize();  
 toggleRelativeSize(); // Initialize relative size fields  
 }  
 </script>  
 {% if error %}  
 <p class="error">{{ error }}</p>  
 {% endif %}  
 {% if results is not none %}  
 <h3>Project Results:</h3>  
 {% if results %}  
 <div class="table-container">  
 <table border="1" cellpadding="5">  
 <tr><th>UUID</th><th>Project Name</th><th>User Name</th><th>Date</th><th>File Location</th><th>Paper Size</th><th>Description</th><th>Associated Scales</th><th class="actions-column">Actions</th></tr>  
 {% for proj in results %}  
 {% if proj and proj.uuid %}  
 <tr>  
 <td>{{ proj.uuid }}</td>  
 <td>{{ proj.project\_name }}</td>  
 <td>{{ proj.user\_name }}</td>  
 <td>{{ proj.date }}</td>  
 <td>{{ proj.file\_location }}</td>  
 <td>{{ proj.paper\_size }}</td>  
 <td>{{ proj.description }}</td>  
 <td>{{ proj.get('associated\_scales', 'N/A') }}</td>  
 <td class="actions-column">  
 {% if proj.view\_file\_path %}  
 <a href="#" onclick="showFileModal('{{ url\_for('view\_file', rel\_path=proj.view\_file\_path) }}','{{ proj.view\_file\_type }}'); return false">View</a>  
 {% else %}  
 <span>No file</span>  
 {% endif %}  
 <a href="#" onclick="copyPath('{{ proj.file\_location.replace('\\', '\\\\')|safe }}'); return false" style="background-color: #27ae60;">Copy Path</a>  
 <form method="post" action="{{ url\_for('delete\_project', uuid=proj.uuid|e) }}" style="display:inline;" onsubmit="return confirm('Are you sure you want to delete this project?');"><button type="submit">Delete</button></form>  
 </td>  
 </tr>  
 {% endif %}  
 {% endfor %}  
 </table>  
 </div>  
 {% else %}  
 <p>No matching projects found.</p>  
 {% endif %}  
 {% endif %}  
 <hr>  
 <h2>All Projects</h2>  
 <div class="table-container">  
 <form method="get" id="projectsFilterForm">  
 <input type="hidden" name="page" value="{{ projects\_current\_page }}">  
 <input type="hidden" name="per\_page" value="{{ projects\_per\_page }}">  
 <table border="1" cellpadding="5">  
 <thead>  
 <tr>  
 <th>UUID <br> <input type="text" name="projects\_uuid\_filter" class="filter-input" value="{{ projects\_filters.uuid\_filter }}" placeholder="Filter UUID"></th>  
 <th>Project Name <br> <input type="text" name="projects\_project\_name\_filter" class="filter-input" value="{{ projects\_filters.project\_name\_filter }}" placeholder="Filter Name"></th>  
 <th>User Name <br> <input type="text" name="projects\_user\_name\_filter" class="filter-input" value="{{ projects\_filters.user\_name\_filter }}" placeholder="Filter User"></th>  
 <th>Date <br>   
 <input type="text" name="projects\_date\_filter" class="filter-input" value="{{ projects\_filters.date\_filter }}" placeholder="Filter Date" style="width: 100%; margin-bottom: 2px;">  
 <input type="text" name="projects\_date\_from\_filter" class="filter-input" value="{{ projects\_filters.date\_from\_filter }}" placeholder="DD/MM/YYYY" style="width: 48%; font-size: 0.8em;" pattern="[0-9]{2}/[0-9]{2}/[0-9]{4}">  
 <input type="text" name="projects\_date\_to\_filter" class="filter-input" value="{{ projects\_filters.date\_to\_filter }}" placeholder="DD/MM/YYYY" style="width: 48%; font-size: 0.8em; margin-left: 2%;" pattern="[0-9]{2}/[0-9]{2}/[0-9]{4}">  
 </th>  
 <th>File Location <br> <input type="text" name="projects\_file\_location\_filter" class="filter-input" value="{{ projects\_filters.file\_location\_filter }}" placeholder="Filter Location"></th>  
 <th>Paper Size <br> <input type="text" name="projects\_paper\_size\_filter" class="filter-input" value="{{ projects\_filters.paper\_size\_filter }}" placeholder="Filter Size"></th>  
 <th>Description</th>  
 <th>Associated Scales <br> <input type="text" name="projects\_associated\_scales\_filter" class="filter-input" value="{{ projects\_filters.associated\_scales\_filter }}" placeholder="Filter Scales"></th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for proj in projects %}  
 {% if proj and proj.uuid %}  
 <tr>  
 <td>{{ proj.uuid }}</td>  
 <td>{{ proj.project\_name }}</td>  
 <td>{{ proj.user\_name }}</td>  
 <td>{{ proj.date }}</td>  
 <td>{{ proj.file\_location }}</td>  
 <td>{{ proj.paper\_size }}</td>  
 <td>{{ proj.description }}</td>  
 <td>{{ proj.get('associated\_scales', 'N/A') }}</td>   
 <td class="actions-column">  
 {% if proj.view\_file\_path %}  
 <a href="#" onclick="showFileModal('{{ url\_for('view\_file', rel\_path=proj.view\_file\_path) }}','{{ proj.view\_file\_type }}'); return false">View</a>  
 {% else %}  
 <span>No file</span>  
 {% endif %}  
 <a href="#" onclick="copyPath('{{ proj.file\_location.replace('\\', '\\\\')|safe }}'); return false" style="background-color: #27ae60;">Copy Path</a>  
 </td>  
 </tr>  
 {% endif %}  
 {% endfor %}  
 </tbody>  
 </table>  
 </form>  
 </div>  
 <div class="pagination">  
 {% if projects\_current\_page > 1 %}  
 <a href="{{ url\_for('index', page=projects\_current\_page - 1, per\_page=projects\_per\_page, \*\*projects\_filters) }}">Previous</a>  
 {% else %}  
 <span class="disabled">Previous</span>  
 {% endif %}  
  
 {% for p in range(1, projects\_total\_pages + 1) %}  
 {% if p == projects\_current\_page %}  
 <span class="current-page">{{ p }}</span>  
 {% else %}  
 <a href="{{ url\_for('index', page=p, per\_page=projects\_per\_page, \*\*projects\_filters) }}">{{ p }}</a>  
 {% endif %}  
 {% endfor %}  
  
 {% if projects\_current\_page < projects\_total\_pages %}  
 <a href="{{ url\_for('index', page=projects\_current\_page + 1, per\_page=projects\_per\_page, \*\*projects\_filters) }}">Next</a>  
 {% else %}  
 <span class="disabled">Next</span>  
 {% endif %}  
 <br>  
 <label>Items per page:  
 <select onchange="window.location.href = '{{ url\_for('index', \*\*projects\_filters) }}' + '&per\_page=' + this.value">  
 {% for size in [5, 10, 20, 50] %}  
 <option value="{{ size }}" {% if projects\_per\_page == size %}selected{% endif %}>{{ size }}</option>  
 {% endfor %}  
 </select>  
 </label>  
 </div>  
  
 <h2>All Areas</h2>  
 <div class="table-container">  
 <form method="get" id="areasFilterForm">  
 <input type="hidden" name="areas\_page" value="{{ areas\_current\_page }}">  
 <input type="hidden" name="areas\_per\_page" value="{{ areas\_per\_page }}">  
 <table border="1" cellpadding="5">  
 <thead>  
 <tr>  
 <th>ID <br> <input type="text" name="areas\_id\_filter" class="filter-input" value="{{ areas\_filters.id\_filter }}" placeholder="Filter ID"></th>  
 <th>Project UUID <br> <input type="text" name="areas\_project\_id\_filter" class="filter-input" value="{{ areas\_filters.project\_id\_filter }}" placeholder="Filter UUID"></th>  
 <th>XMin <br> <input type="text" name="areas\_xmin\_filter" class="filter-input" value="{{ areas\_filters.xmin\_filter }}" placeholder="Filter XMin"></th>  
 <th>YMin <br> <input type="text" name="areas\_ymin\_filter" class="filter-input" value="{{ areas\_filters.ymin\_filter }}" placeholder="Filter YMin"></th>  
 <th>XMax <br> <input type="text" name="areas\_xmax\_filter" class="filter-input" value="{{ areas\_filters.xmax\_filter }}" placeholder="Filter XMax"></th>  
 <th>YMax <br> <input type="text" name="areas\_ymax\_filter" class="filter-input" value="{{ areas\_filters.ymax\_filter }}" placeholder="Filter YMax"></th>  
 <th>Scale <br> <input type="text" name="areas\_scale\_filter" class="filter-input" value="{{ areas\_filters.scale\_filter }}" placeholder="Filter Scale"></th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for area in areas %}  
 <tr>  
 <td>{{ area.id }}</td>  
 <td>{{ area.project\_id }}</td>  
 <td>{{ area.xmin }}</td>  
 <td>{{ area.ymin }}</td>  
 <td>{{ area.xmax }}</td>  
 <td>{{ area.ymax }}</td>  
 <td>{{ area.scale.replace('Scale:', '').strip() if area.scale else '' }}</td>  
 <td class="actions-column">  
 <a href="#" onclick="showFileModalOrNoFiles({{ area.project\_all\_files|tojson|safe }}); return false">View Project</a>  
 <a href="#" onclick="copyPath('{{ area.project\_file\_location.replace('\\', '\\\\')|safe }}'); return false" style="background-color: #27ae60;">Copy Project Path</a>  
 <button type="button" onclick="copyTopRight('{{ area.xmax }}', '{{ area.ymax }}')">Copy Top Right</button>  
 <button type="button" onclick="copyBottomLeft('{{ area.xmin }}', '{{ area.ymin }}')">Copy Bottom Left</button>  
 </td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
 </form>  
 </div>  
 <div class="pagination">  
 {% if areas\_current\_page > 1 %}  
 <a href="{{ url\_for('index', areas\_page=areas\_current\_page - 1, areas\_per\_page=areas\_per\_page, \*\*areas\_filters) }}">Previous</a>  
 {% else %}  
 <span class="disabled">Previous</span>  
 {% endif %}  
  
 {% for p in range(1, areas\_total\_pages + 1) %}  
 {% if p == areas\_current\_page %}  
 <span class="current-page">{{ p }}</span>  
 {% else %}  
 <a href="{{ url\_for('index', areas\_page=p, areas\_per\_page=areas\_per\_page, \*\*areas\_filters) }}">{{ p }}</a>  
 {% endif %}  
 {% endfor %}  
  
 {% if areas\_current\_page < areas\_total\_pages %}  
 <a href="{{ url\_for('index', areas\_page=areas\_current\_page + 1, areas\_per\_page=areas\_per\_page, \*\*areas\_filters) }}">Next</a>  
 {% else %}  
 <span class="disabled">Next</span>  
 {% endif %}  
 <br>  
 <label>Items per page:  
 <select onchange="window.location.href = '{{ url\_for('index', \*\*areas\_filters) }}' + '&areas\_per\_page=' + this.value">  
 {% for size in [5, 10, 20, 50] %}  
 <option value="{{ size }}" {% if areas\_per\_page == size %}selected{% endif %}>{{ size }}</option>  
 {% endfor %}  
 </select>  
 </label>  
 </div>  
 <div id="fileModal" style="display:none; position:fixed; top:0; left:0; width:100vw; height:100vh; background:rgba(0,0,0,0.8); z-index:9999; align-items:center; justify-content:center;">  
 <div id="fileModalContent" style="position:relative; background:#fff; padding:20px; border-radius:8px; max-width:90vw; max-height:90vh; overflow:auto;">  
 <button onclick="closeFileModal()" style="position:absolute; top:10px; right:10px; z-index:10000;">Close</button>  
 <div id="fileModalBody"></div>  
 </div>  
 </div>  
   
 <!-- Gallery Modal for All Files -->  
 <div id="galleryModal" style="display:none; position:fixed; top:0; left:0; width:100vw; height:100vh; background:rgba(0,0,0,0.9); z-index:10000; align-items:center; justify-content:center;">  
 <div id="galleryModalContent" style="position:relative; background:#fff; padding:20px; border-radius:8px; max-width:95vw; max-height:95vh; overflow:hidden; display:flex; flex-direction:column;">  
 <div style="display:flex; justify-content:space-between; align-items:center; margin-bottom:15px;">  
 <h3 id="galleryTitle" style="margin:0; color:#333;">Project Files</h3>  
 <button onclick="closeGalleryModal()" style="background:#e74c3c; color:white; border:none; padding:8px 12px; border-radius:4px; cursor:pointer;">Close</button>  
 </div>  
   
 <div id="galleryContainer" style="flex:1; display:flex; align-items:center; justify-content:center; position:relative;">  
 <!-- Navigation Arrows -->  
 <button id="prevBtn" onclick="previousFile()" style="position:absolute; left:10px; top:50%; transform:translateY(-50%); background:rgba(0,0,0,0.7); color:white; border:none; padding:15px 10px; border-radius:50%; cursor:pointer; font-size:18px; z-index:10001;">‹</button>  
 <button id="nextBtn" onclick="nextFile()" style="position:absolute; right:10px; top:50%; transform:translateY(-50%); background:rgba(0,0,0,0.7); color:white; border:none; padding:15px 10px; border-radius:50%; cursor:pointer; font-size:18px; z-index:10001;">›</button>  
   
 <!-- File Display Area -->  
 <div id="galleryFileDisplay" style="max-width:90%; max-height:80vh; display:flex; align-items:center; justify-content:center;">  
 <!-- Content will be loaded here -->  
 </div>  
 </div>  
   
 <!-- File Info and Navigation -->  
 <div style="display:flex; justify-content:space-between; align-items:center; margin-top:15px; padding:10px; background:#f8f9fa; border-radius:4px;">  
 <div id="fileInfo" style="flex:1;">  
 <div id="fileName" style="font-weight:bold; margin-bottom:5px;"></div>  
 <div id="fileDate" style="font-size:0.9em; color:#666;"></div>  
 </div>  
 <div id="fileCounter" style="text-align:center; font-weight:bold; color:#333;"></div>  
 <div id="fileType" style="background:#3498db; color:white; padding:4px 8px; border-radius:12px; font-size:0.8em;"></div>  
 </div>  
 </div>  
 </div>  
   
  
 <script>  
 function showFileModal(url, type) {  
 console.log('[DEBUG] showFileModal called with url:', url, 'type:', type);  
 var modal = document.getElementById('fileModal');  
 var body = document.getElementById('fileModalBody');  
 if (type === 'pdf') {  
 body.innerHTML = '<iframe src="' + url + '" width="800" height="600" style="border:none;"></iframe>';  
 } else if (type === 'img') {  
 body.innerHTML = '<img src="' + url + '" style="max-width:80vw; max-height:80vh; display:block; margin:auto;" />';  
 }  
 modal.style.display = 'flex';  
 }  
 function closeFileModal() {  
 var modal = document.getElementById('fileModal');  
 var body = document.getElementById('fileModalBody');  
 body.innerHTML = '';  
 modal.style.display = 'none';  
 }  
   
 // Gallery modal variables  
 var currentFiles = [];  
 var currentFileIndex = 0;  
   
 function showAllFilesModal(uuid) {  
 try {  
 // Get the clicked element and its data-files attribute  
 var clickedElement = event.target;  
 var filesJson = clickedElement.getAttribute('data-files');  
   
 currentFiles = JSON.parse(filesJson);  
 currentFileIndex = 0;  
   
 var modal = document.getElementById('galleryModal');  
 var title = document.getElementById('galleryTitle');  
   
 title.textContent = 'Project Files (' + currentFiles.length + ' files)';  
 modal.style.display = 'flex';  
   
 displayCurrentFile();  
 } catch (error) {  
 console.error('Error parsing files data:', error);  
 alert('Error loading files. Please try again.');  
 }  
 }  
   
 function closeGalleryModal() {  
 var modal = document.getElementById('galleryModal');  
 modal.style.display = 'none';  
 currentFiles = [];  
 currentFileIndex = 0;  
 }  
   
 function displayCurrentFile() {  
 if (currentFiles.length === 0) return;  
   
 var file = currentFiles[currentFileIndex];  
 var display = document.getElementById('galleryFileDisplay');  
 var fileName = document.getElementById('fileName');  
 var fileDate = document.getElementById('fileDate');  
 var fileCounter = document.getElementById('fileCounter');  
 var fileType = document.getElementById('fileType');  
 var prevBtn = document.getElementById('prevBtn');  
 var nextBtn = document.getElementById('nextBtn');  
   
 // Update file info  
 fileName.textContent = file.filename;  
 fileDate.textContent = new Date(file.ctime \* 1000).toLocaleString();  
 fileCounter.textContent = (currentFileIndex + 1) + ' / ' + currentFiles.length;  
 fileType.textContent = file.type.toUpperCase();  
   
 // Update navigation buttons  
 prevBtn.style.display = currentFileIndex > 0 ? 'block' : 'none';  
 nextBtn.style.display = currentFileIndex < currentFiles.length - 1 ? 'block' : 'none';  
   
 // Generate URL dynamically  
 var fileUrl = '/view\_file/' + encodeURIComponent(file.rel\_path);  
   
 // Display file content  
 if (file.type === 'pdf') {  
 display.innerHTML = '<iframe src="' + fileUrl + '" width="800" height="600" style="border:none; max-width:100%; max-height:100%;"></iframe>';  
 } else {  
 display.innerHTML = '<img src="' + fileUrl + '" style="max-width:100%; max-height:100%; object-fit:contain;" alt="' + file.filename + '">';  
 }  
 }  
   
 function previousFile() {  
 if (currentFileIndex > 0) {  
 currentFileIndex--;  
 displayCurrentFile();  
 }  
 }  
   
 function nextFile() {  
 if (currentFileIndex < currentFiles.length - 1) {  
 currentFileIndex++;  
 displayCurrentFile();  
 }  
 }  
   
 // Keyboard navigation  
 document.addEventListener('keydown', function(event) {  
 if (document.getElementById('galleryModal').style.display === 'flex') {  
 if (event.key === 'ArrowLeft') {  
 previousFile();  
 } else if (event.key === 'ArrowRight') {  
 nextFile();  
 } else if (event.key === 'Escape') {  
 closeGalleryModal();  
 }  
 }  
 });  
  
 function showFileModalOrNoFiles(files) {  
 console.log('[DEBUG] showFileModalOrNoFiles called with files:', files);  
 if (!files || files.length === 0) {  
 var modal = document.getElementById('fileModal');  
 var body = document.getElementById('fileModalBody');  
 body.innerHTML = '<div style="text-align:center; padding:40px; font-size:1.2em; color:#888;">No files available for this project.</div>';  
 modal.style.display = 'flex';  
 } else if (files.length === 1) {  
 var file = files[0];  
 var url = '/view\_file/' + encodeURIComponent(file.rel\_path);  
 console.log('[DEBUG] Single file, url:', url, 'type:', file.type);  
 showFileModal(url, file.type);  
 } else {  
 // Multiple files: open gallery  
 currentFiles = files;  
 currentFileIndex = 0;  
 var modal = document.getElementById('galleryModal');  
 var title = document.getElementById('galleryTitle');  
 title.textContent = 'Project Files (' + currentFiles.length + ' files)';  
 modal.style.display = 'flex';  
 displayCurrentFile();  
 }  
 }  
  
 function copyTopRight(xmax, ymax) {  
 var str = xmax + '/' + ymax;  
 var textarea = document.createElement('textarea');  
 textarea.value = str;  
 document.body.appendChild(textarea);  
 textarea.select();  
 document.execCommand('copy');  
 document.body.removeChild(textarea);  
 showCopyNotification('Top Right copied: ' + str);  
 }  
 function copyBottomLeft(xmin, ymin) {  
 var str = xmin + '/' + ymin;  
 var textarea = document.createElement('textarea');  
 textarea.value = str;  
 document.body.appendChild(textarea);  
 textarea.select();  
 document.execCommand('copy');  
 document.body.removeChild(textarea);  
 showCopyNotification('Bottom Left copied: ' + str);  
 }  
 function showCopyNotification(msg) {  
 var notification = document.createElement('div');  
 notification.textContent = msg;  
 notification.style.cssText = 'position: fixed; top: 20px; right: 20px; background: #27ae60; color: white; padding: 10px 15px; border-radius: 5px; z-index: 10000; font-size: 14px;';  
 document.body.appendChild(notification);  
 setTimeout(function() {  
 if (notification.parentNode) {  
 notification.parentNode.removeChild(notification);  
 }  
 }, 2000);  
 }  
 </script>  
   
 <!-- Footer with Logo and Copyright -->  
 <div class="header-footer">  
 <div class="logo-section">  
 <div class="logo">  
 <img src="{{ url\_for('static', filename='rocket.jpg') }}" alt="Rocket Logo" style="width: 36px; height: 36px; object-fit: contain; border-radius: 50%; background: white;">  
 </div>  
 <div class="company-info">  
 <strong>ARCgis PRO DataBase</strong><br>  
 <span style="font-size: 11px;">פרוייקט שימור הידע של (נוסיף שיתאפשר)</span><br>  
 <span class="copyright">@Rocket Team Production</span>  
 </div>  
 </div>  
  
 </div>  
</body>  
</html>

--------------------------------------------------