YOLOv8 Object Detection API with Custom Datasets

Objective: Develop a server-side application using Python and Flask to integrate YOLOv8 with custom datasets. The application must have two POST endpoints, one for receiving object detection data in JSON format and another for receiving an annotated image with detected objects.

Requirements:

1. Use Python 3.x and Flask for server-side application development.
2. Integrate YOLOv8 object detection model using custom datasets.
3. Implement two POST API endpoints:

Endpoint 1:

- URL: /api/detect_objects
- Method: POST
- Input: Multipart form-data containing an image file (supported formats: JPEG, PNG)
- Output: JSON response containing detected objects with their xmin, ymin, xmax, ymax, confidence, class, and name.

Example JSON response:


```
"result": [
{
"xmin": 721.4908447266,
"ymin": 329.434753418,
"xmax": 771.1446533203,
"ymax": 372.4448242188,
"confidence": 0.0880783051,
"class": 4,
"name": "object_1"
},
{
"xmin": 659.9022827148,
"ymin": 232.1421203613,
"xmax": 704.4338378906,
"ymax": 375.7359008789,
"confidence": 0.0715641975,
"class": 3,
"name": "object_2"
}
]
```

Endpoint 2:

- URL: /api/annotate_image
- Method: POST
- Input: Multipart form-data containing an image file (supported formats: JPEG, PNG)
- Output: Image file (same format as input) with detected objects annotated (e.g., bounding boxes and labels).

Deliverables:

1. Source code of the server-side application, including all dependencies (requirements.txt) and instructions for deployment.
2. Test script with example images to demonstrate the functionality of both API endpoints.
3. Documentation on how to use the API, including example requests and responses.