Name: Vishal Ranka
USN: 1BM21IS205

Step1 : Flask app (app.py)

app.py

assignment_1 > app.py

```python
from flask import Flask, request, jsonify
from pymongo import MongoClient
from bson.objectid import ObjectId

app = Flask(__name__)

# MongoDB configuration
client = MongoClient('mongodb://mongo:27017/')
db = client['mydatabase']
collection = db['items']

# Routes
@app.route('/items', methods=['POST']   )
def create_item():
    name = request.json['name']
    item_id = collection.insert_one({'name': name}).inserted_id
    return jsonify({'message': 'Item created', 'id': str(item_id)}), 201

@app.route('/items', methods=['GET'])
def get_items():
    items = collection.find()
    return jsonify([{'id': str(item['_id']), 'name': item['name']} for item in items])

@app.route('/items/<id>', methods=['PUT'])
def update_item(id):
    collection.update_one({'_id': ObjectId(id)}, {'$set': {'name': request.json['name']}})
    return jsonify({'message': 'Item updated'})

@app.route('/items/<id>', methods=['DELETE'])
def delete_item(id):
    collection.delete_one({'_id': ObjectId(id)})
    return jsonify({'message': 'Item deleted'})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

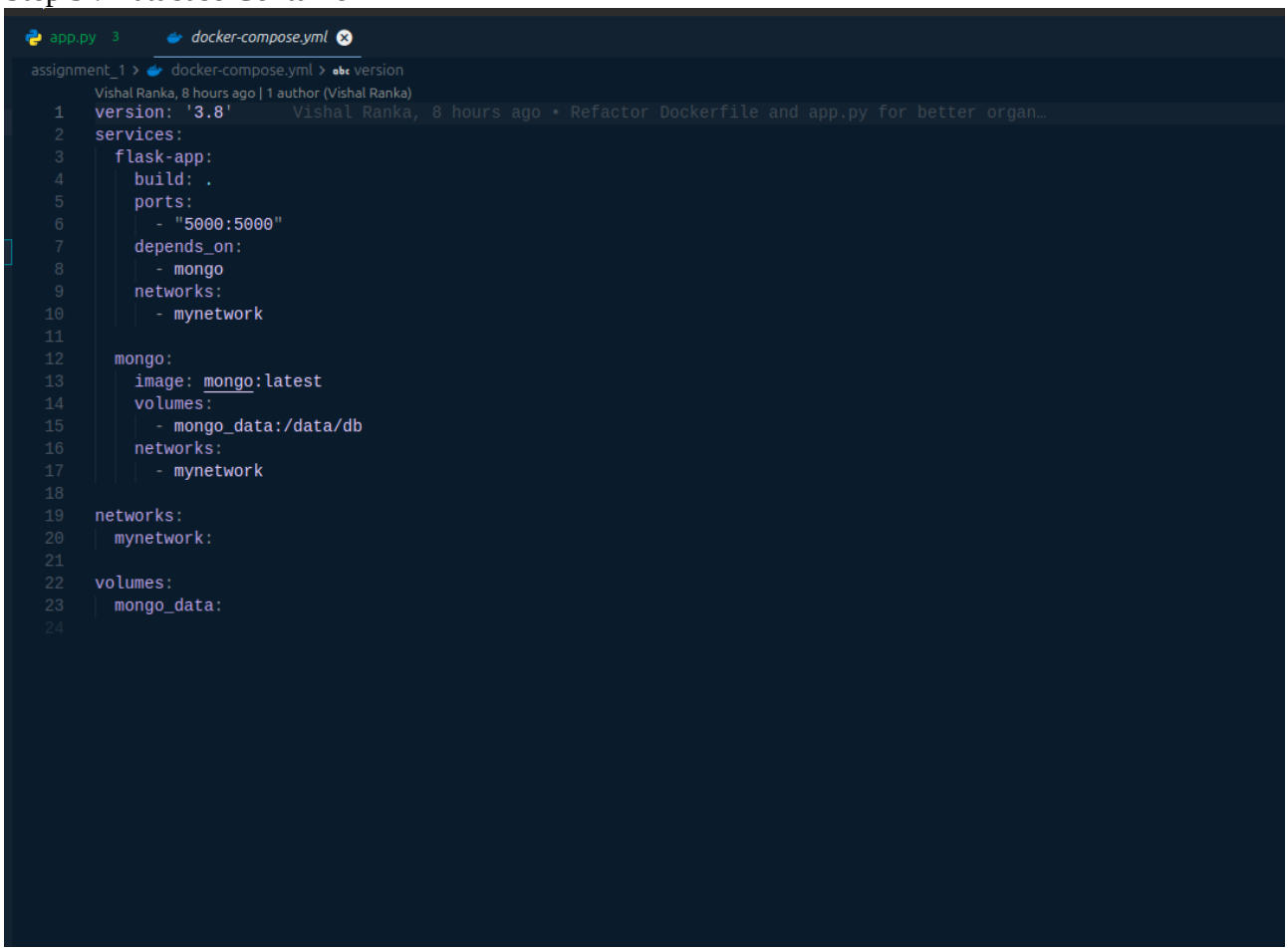
Step 2 : Containerize

app.py

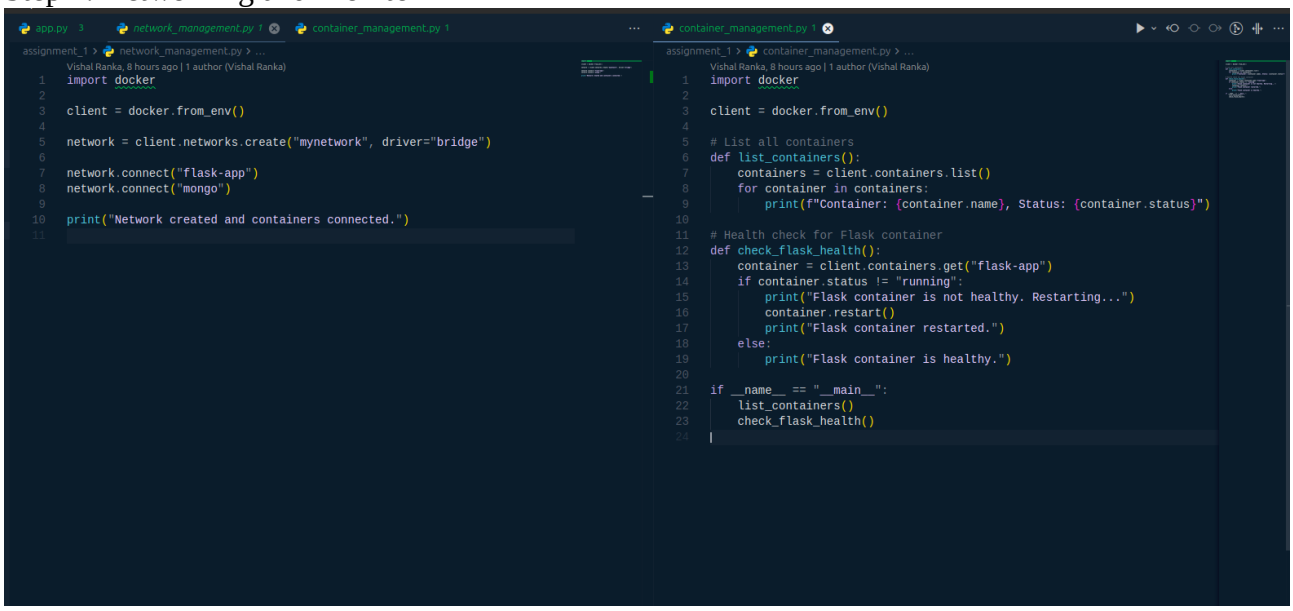
assignment_1 > app.py

```python
from flask import Flask, request, jsonify
from pymongo import MongoClient
from bson.objectid import ObjectId

app = Flask(__name__)

# MongoDB configuration
client = MongoClient('mongodb://mongo:27017/')
db = client['mydatabase']
collection = db['items']

# Routes
@app.route('/items', methods=['POST']   )
def create_item():
    name = request.json['name']
    item_id = collection.insert_one({'name': name}).inserted_id
    return jsonify({'message': 'Item created', 'id': str(item_id)}), 201

@app.route('/items', methods=['GET'])
def get_items():
    items = collection.find()
    return jsonify([{'id': str(item['_id']), 'name': item['name']} for item in items])

@app.route('/items/<id>', methods=['PUT'])
def update_item(id):
    collection.update_one({'_id': ObjectId(id)}, {'$set': {'name': request.json['name']}})
    return jsonify({'message': 'Item updated'})

@app.route('/items/<id>', methods=['DELETE'])
def delete_item(id):
    collection.delete_one({'_id': ObjectId(id)})
    return jsonify({'message': 'Item deleted'})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Step 3 : Database Container



Step 4: Networking and Monitor



```python
import docker

client = docker.from_env()

network = client.networks.create("mynetwork", driver="bridge")

network.connect("flask-app")
network.connect("mongo")

print("Network created and containers connected.")
```

```python
import docker

client = docker.from_env()

# List all containers
def list_containers():
    containers = client.containers.list()
    for container in containers:
        print(f"Container: {container.name}, Status: {container.status}")

# Health check for Flask container
def check_flask_health():
    container = client.containers.get("flask-app")
    if container.status != "running":
        print("Flask container is not healthy. Restarting...")
        container.restart()
        print("Flask container restarted.")
    else:
        print("Flask container is healthy.")

if __name__ == "__main__":
    list_containers()
    check_flask_health()
```