

Name: Vishal Ranka

USN: 1BM21IS205

Step1 : Flask app (app.py)

```
app.py
assignment_1 > app.py
1  from flask import Flask, request, jsonify
2  from pymongo import MongoClient
3  from bson.objectid import ObjectId
4
5  app = Flask(__name__)
6
7  # MongoDB configuration
8  client = MongoClient('mongodb://mongo:27017/')
9  db = client['mydatabase']
10 collection = db['items']
11
12 # Routes
13 @app.route('/items', methods=['POST'])
14 def create_item():
15     name = request.json['name']
16     item_id = collection.insert_one({'name': name}).inserted_id
17     return jsonify({'message': 'Item created', 'id': str(item_id)}), 201
18
19 @app.route('/items', methods=['GET'])
20 def get_items():
21     items = collection.find()
22     return jsonify([{'id': str(item['_id']), 'name': item['name']} for item in items])
23
24 @app.route('/items/<id>', methods=['PUT'])
25 def update_item(id):
26     collection.update_one({'_id': ObjectId(id)}, {'$set': {'name': request.json['name']}})
27     return jsonify({'message': 'Item updated'})
28
29 @app.route('/items/<id>', methods=['DELETE'])
30 def delete_item(id):
31     collection.delete_one({'_id': ObjectId(id)})
32     return jsonify({'message': 'Item deleted'})
33
34 if __name__ == '__main__':
35     app.run(host='0.0.0.0', port=5000)
36
```

Step 2 : Containerize

```
app.py
assignment_1 > app.py
1  from flask import Flask, request, jsonify
2  from pymongo import MongoClient
3  from bson.objectid import ObjectId
4
5  app = Flask(__name__)
6
7  # MongoDB configuration
8  client = MongoClient('mongodb://mongo:27017/')
9  db = client['mydatabase']
10 collection = db['items']
11
12 # Routes
13 @app.route('/items', methods=['POST'])
14 def create_item():
15     name = request.json['name']
16     item_id = collection.insert_one({'name': name}).inserted_id
17     return jsonify({'message': 'Item created', 'id': str(item_id)}), 201
18
19 @app.route('/items', methods=['GET'])
20 def get_items():
21     items = collection.find()
22     return jsonify([{'id': str(item['_id']), 'name': item['name']} for item in items])
23
24 @app.route('/items/<id>', methods=['PUT'])
25 def update_item(id):
26     collection.update_one({'_id': ObjectId(id)}, {'$set': {'name': request.json['name']}})
27     return jsonify({'message': 'Item updated'})
28
29 @app.route('/items/<id>', methods=['DELETE'])
30 def delete_item(id):
31     collection.delete_one({'_id': ObjectId(id)})
32     return jsonify({'message': 'Item deleted'})
33
34 if __name__ == '__main__':
35     app.run(host='0.0.0.0', port=5000)
36
```

Step 3 : Database Container

```
app.py 3 docker-compose.yml ✕
assignment_1 > docker-compose.yml > version
Vishal Ranka, 8 hours ago | 1 author (Vishal Ranka)
1 version: '3.8' Vishal Ranka, 8 hours ago • Refactor Dockerfile and app.py for better organ...
2 services:
3   flask-app:
4     build: .
5     ports:
6       - "5000:5000"
7     depends_on:
8       - mongo
9     networks:
10      - mynetwork
11
12   mongo:
13     image: mongo:latest
14     volumes:
15       - mongo_data:/data/db
16     networks:
17       - mynetwork
18
19 networks:
20   mynetwork:
21
22 volumes:
23   mongo_data:
24
```

Step 4: Networking and Monitor

```
assignment_1 > network_management.py > ...
1 import docker
2
3 client = docker.from_env()
4
5 network = client.networks.create("mynetwork", driver="bridge")
6
7 network.connect("flask-app")
8 network.connect("mongo")
9
10 print("Network created and containers connected.")
11
```

```
assignment_1 > container_management.py > ...
1 import docker
2
3 client = docker.from_env()
4
5 # List all containers
6 def list_containers():
7     containers = client.containers.list()
8     for container in containers:
9         print(f"Container: {container.name}, Status: {container.status}")
10
11 # Health check for Flask container
12 def check_flask_health():
13     container = client.containers.get("flask-app")
14     if container.status != "running":
15         print("Flask container is not healthy. Restarting...")
16         container.restart()
17         print("Flask container restarted.")
18     else:
19         print("Flask container is healthy.")
20
21 if __name__ == "__main__":
22     list_containers()
23     check_flask_health()
24
```