Name: Mrunali Katta

ID: 017516785

Github repository link: https://github.com/mrunalikatta1998/DATA236-Distributed-Systems-for-Data-Engineering/tree/main/hw11/Homework11

Homework 11

Q1. Shipping Microservice (5 Marks)

Build a shipping microservice that listens to the “order-confirmed” topic and creates a shipping record. (Refer the demo code for orders topic).

Requirements:

- Connect to Kafka as a consumer.
- Subscribe to the topic: "order-confirmed"
- Save record to db with tracking id and status
  - Generate a tracking Id: (e.g SHIP-XYZ)
  - Set the status to ‘Pending’
  - Save the record

NOTE: The shipping schema should consist of

itemId: String

itemName: String

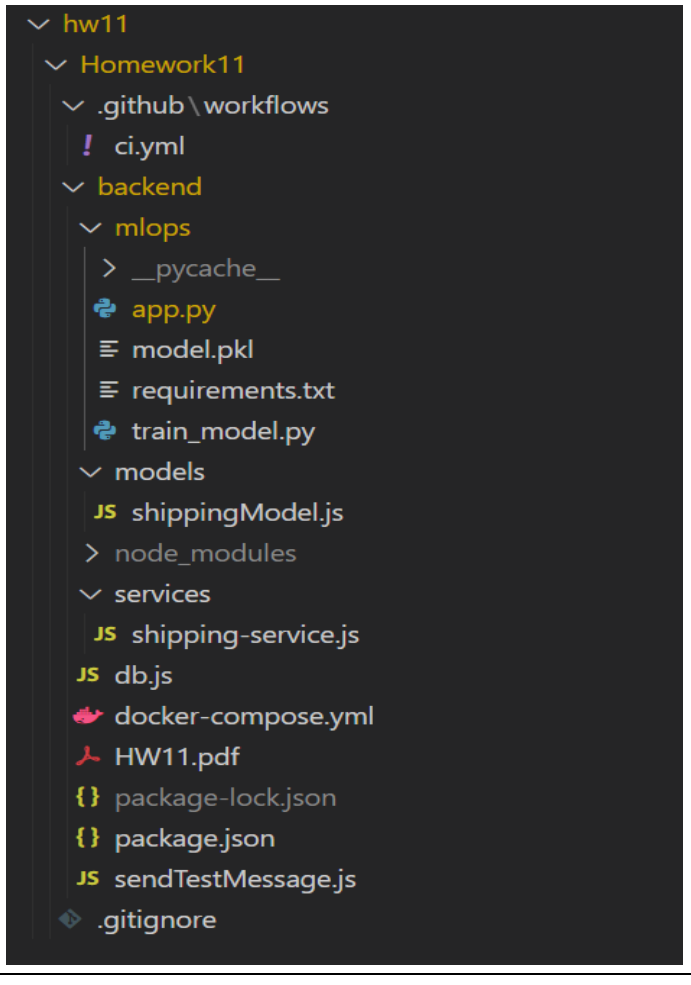
Quantity: Number

trackingId: String

status: { type: String, default: "pending" }

createdAt: { type: Date, default: Date.now }

➔ Cs code folder structure



Homework11\backend\services\shipping-service.js

```
const { Kafka } = require("kafkajs");
const connectDB = require("../db");
const Shipping = require("../models/shippingModel");
const { v4: uuidv4 } = require("uuid");

const kafka = new Kafka({ clientId: "shipping", brokers: ["localhost:9092"] });
const consumer = kafka.consumer({ groupId: "shipping-group" });

async function start() {
  await connectDB();
  await consumer.connect();
  await consumer.subscribe({ topic: "order-confirmed", fromBeginning: true });

  await consumer.run({
    eachMessage: async ({ message }) => {
      const msg = JSON.parse(message.value.toString());
      console.log("Shipping service received:", msg);

      const trackingId = "SHIP-" + uuidv4().slice(0, 6).toUpperCase();

      const newShipping = await Shipping.create({
        itemId: msg.itemId,
        itemName: msg.itemName,
        quantity: msg.quantity,
        trackingId,
        status: "Pending"
      });

      console.log("Shipping record created:", newShipping);
    },
  });
}

start();
```

Shipping microservice running

```
D:\Distributed systems\hw11\Homework11\backend>node services/shipping-service.js
MongoDB connected
{"level":"INFO","timestamp":"2025-04-23T03:54:49.451Z","logger":"kafkajs","message":"[Consumer] Starting","groupId":"shi
pping-group"}
{"level":"INFO","timestamp":"2025-04-23T03:55:12.939Z","logger":"kafkajs","message":"[ConsumerGroup] Consumer has joined
 the group","groupId":"shipping-group","memberId":"shipping-d1ccdc81-0523-40ff-bfe1-5c85686362d7","leaderId":"shipping-d
1ccdc81-0523-40ff-bfe1-5c85686362d7","isLeader":true,"memberAssignment":{"order-confirmed":[0]},"groupProtocol":"RoundRo
binAssigner","duration":23486}
Shipping service received: { itemId: '123', itemName: 'Toy Rocket', quantity: 2 }
Shipping record created: {
  itemId: '123',
  itemName: 'Toy Rocket',
  quantity: 2,
  trackingId: 'SHIP-99E572',
  status: 'Pending',
  _id: new ObjectId('680864ccc5de112ad8c7f720'),
  createdAt: 2025-04-23T03:55:56.682Z,
  __v: 0
}
```

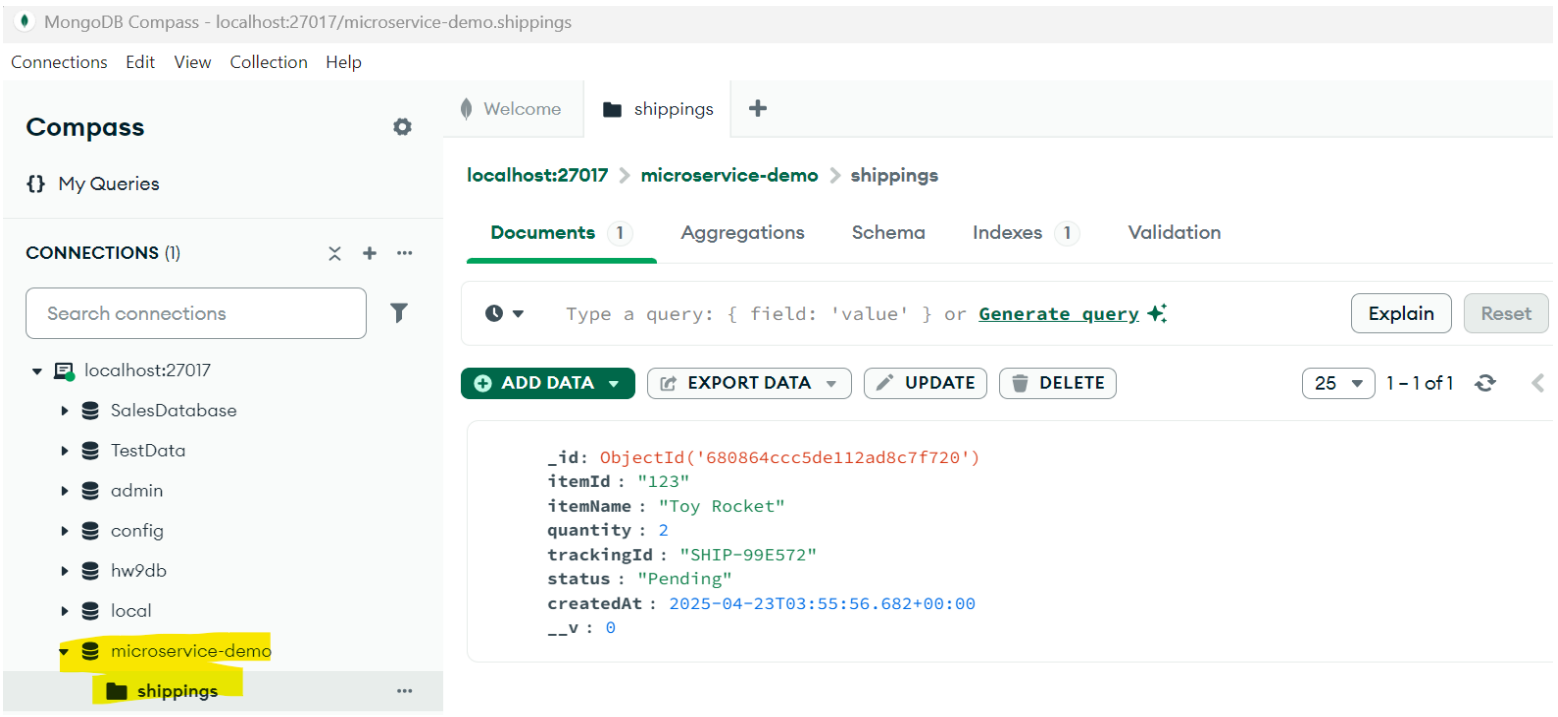
Test message sent to Kafka

```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\Distributed systems\hw11\Homework11\backend>node sendTestMessage.js
{"level":"WARN","timestamp":"2025-04-23T03:55:56.586Z","logger":"kafkajs","message":"KafkaJS v2.0.0 switched default par
titioner. To retain the same partitioning behavior as in previous versions, create the producer with the option \"create
Partitioner: Partitioners.LegacyPartitioner\". See the migration guide at https://kafka.js.org/docs/migration-guide-v2.0
.0#producer-new-default-partitioner for details. Silence this warning by setting the environment variable \"KAFKAJS_NO_P
ARTITIONER_WARNING=1\""}
Test message sent to 'order-confirmed'

D:\Distributed systems\hw11\Homework11\backend>
```

Shipping record saved in MongoDB



Q2. ML-OPS (5 Marks)

Create a mini-MLOps pipeline using your own model and deploy a FastAPI-based prediction endpoint.

● Train a Model - Use a simple dataset like Iris, Wine, or your own mini regression/classification problem, Save the model as a .pkl file using joblib

● Serve Using FastAPI - Build an API with /predict endpoint using FastAPI. Accept input as JSON and return the predicted output

● Set Up GitHub Repo- Upload your project files: app.py, model.pkl, requirements.txt, etc.

● Add CI Workflow - Create .github/workflows/ci.yml. It should install dependencies, run the FastAPI script or sanity check model load.

Submit the screenshot of your CI workflow.yml and the actions page of your GitHub repo with the pipeline successful.

➔ Homework11\backend\mlops\train_model.py **(Iris model training)**

```python
from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import joblib

# Load dataset
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.2, random_state=42
)

# Train model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Save model
joblib.dump(model, "model.pkl")

print("Model trained and saved as model.pkl")
```

Model trained successfully as saved as .pkl file

```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

D:\Distributed systems\hw11\Homework11\backend\mlops>python train_model.py
Model trained and saved as model.pkl
```

Homework11\backend\mlops\app.py

```python
from fastapi import FastAPI
from pydantic import BaseModel
import joblib
import numpy as np

# Load model
model = joblib.load("model.pkl")

# Define input schema
class IrisInput(BaseModel):
    sepal_length: float
    sepal_width: float
    petal_length: float
    petal_width: float

app = FastAPI()
@app.post("/predict")
def predict(input: IrisInput):
    data = np.array([[input.sepal_length, input.sepal_width, input.petal_length,
input.petal_width]])
    prediction = model.predict(data)
    return {"prediction": int(prediction[0])}
```

FastAPI prediction response

```
D:\Distributed systems\hw11\Homework11\backend\mlops>uvicorn app:app --reload
INFO:     Will watch for changes in these directories: ['D:\\Distributed systems\\hw11\\Homework11\\backend\\mlops']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [23656] using StatReload
INFO:     Started server process [5740]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:52745 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:52745 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:52768 - "POST /predict HTTP/1.1" 200 OK
```

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "sepal_length": 5.1,
  "sepal_width": 3.5,
  "petal_length": 1.4,
  "petal_width": 0.2
}
'
```

Request URL

```
http://127.0.0.1:8000/predict
```

Server response

Code	Details
200	Response body

```
{
  "prediction": 0
}
```

Download

Response headers

```
content-length: 16
content-type: application/json
date: Wed,23 Apr 2025 04:26:19 GMT
server: uvicorn
```

Homework11\.github\workflows\ci.yml

```yaml
name: MLOps CI Workflow
# Triggering CI

on:
  push:
  pull_request:

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.10'

    - name: Install dependencies
      run: |
        cd hw11/Homework11/backend/mlops
        pip install -r requirements.txt

    - name: Sanity check - Load model
      run: |
        cd hw11/Homework11/backend/mlops
        python -c "import joblib; joblib.load('model.pkl'); print('☑ Model loaded')"
```

MLOps CI Workflow – Success