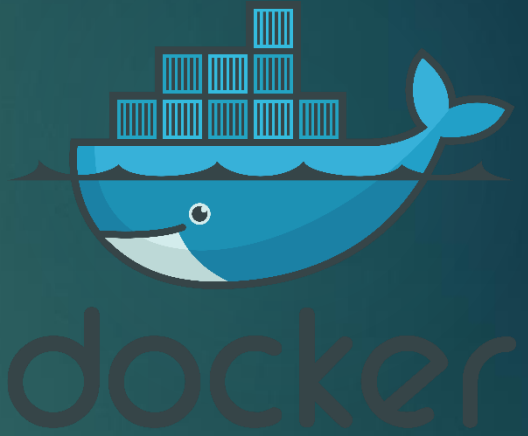
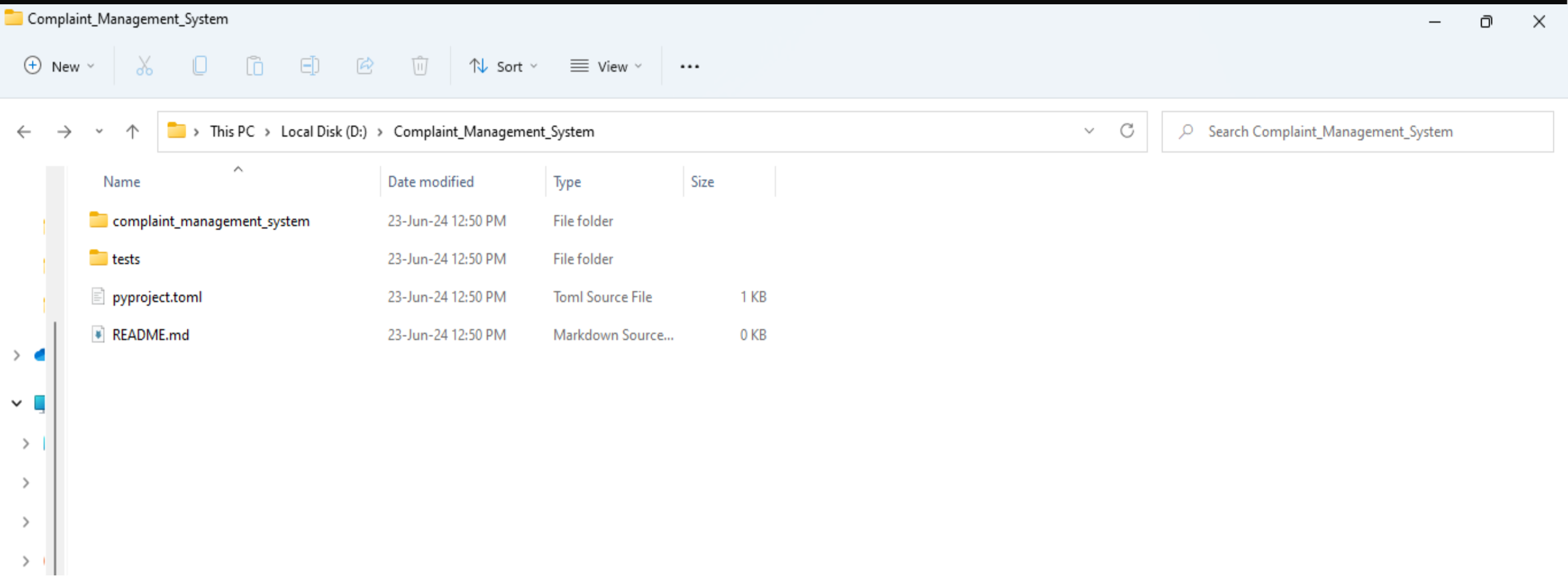
COMPALINT MANAGEMENT SYSTEM
FINAL TERM Project

FastAPI, Kafka, Kong, Streamlit with Docker



Submitted by: TAJ AHMED (PDG DS&AI BV)

```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

D:\>poetry new Complaint_Management_System
Created package complaint_management_system in Complaint_Management_System

D:\>
```



```
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

D:\Complaint_Management_System>poetry shell
Creating virtualenv complaint-management-system-uiWtdocs-py3.12 in C:\Users\HP\AppData\Local\pypoetry\Cache\virtualenvs
Spawning shell within C:\Users\HP\AppData\Local\pypoetry\Cache\virtualenvs\complaint-management-system-uiWtdocs-py3.12

D:\Complaint_Management_System>()

(complaint-management-system-py3.12) D:\Complaint_Management_System>poetry install
Updating dependencies
Resolving dependencies... (0.1s)

Writing lock file

Installing the current project: complaint-management-system (0.1.0)

(complaint-management-system-py3.12) D:\Complaint_Management_System>
```

Complaint_Management_System

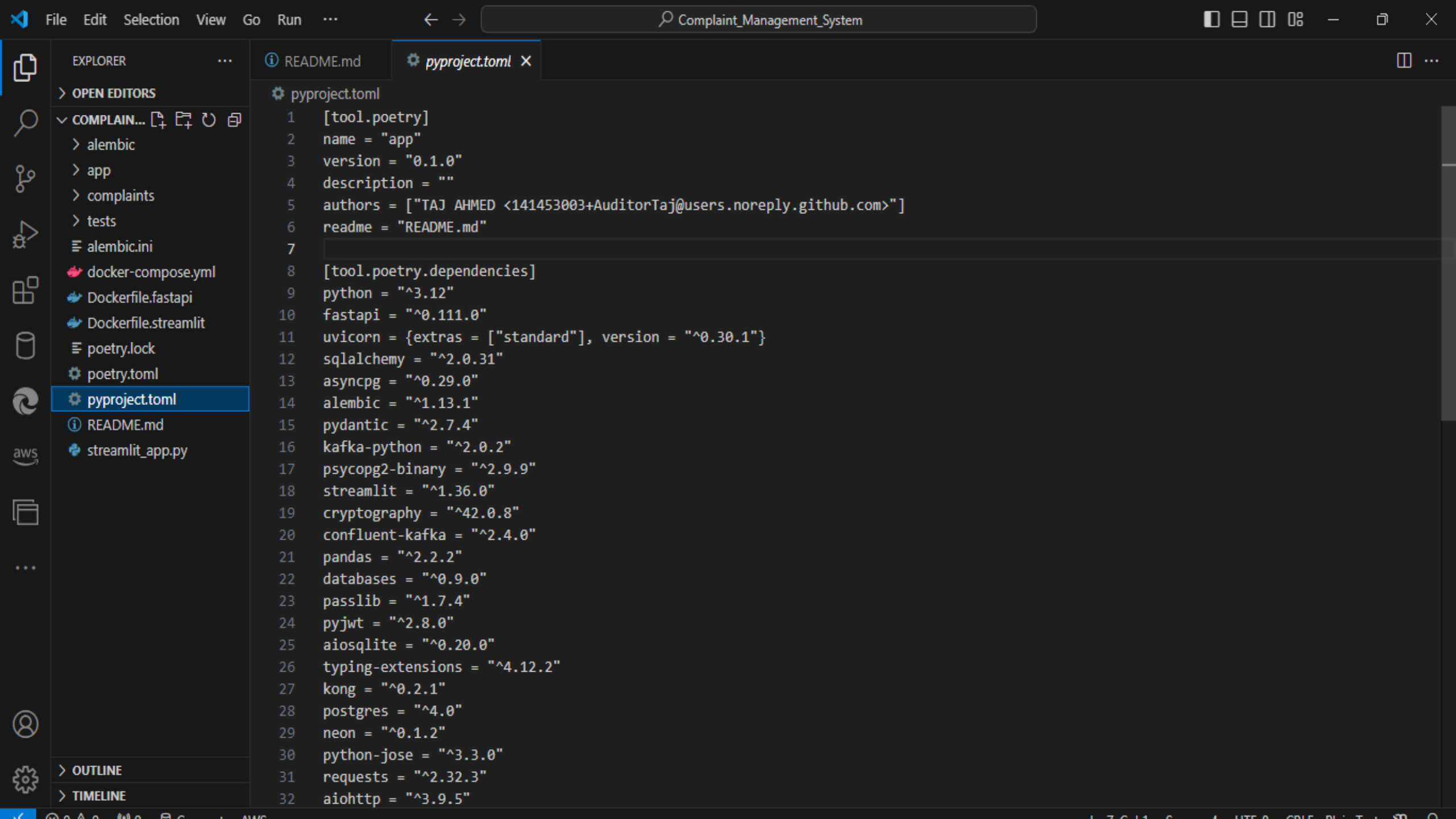
New | Sort | View

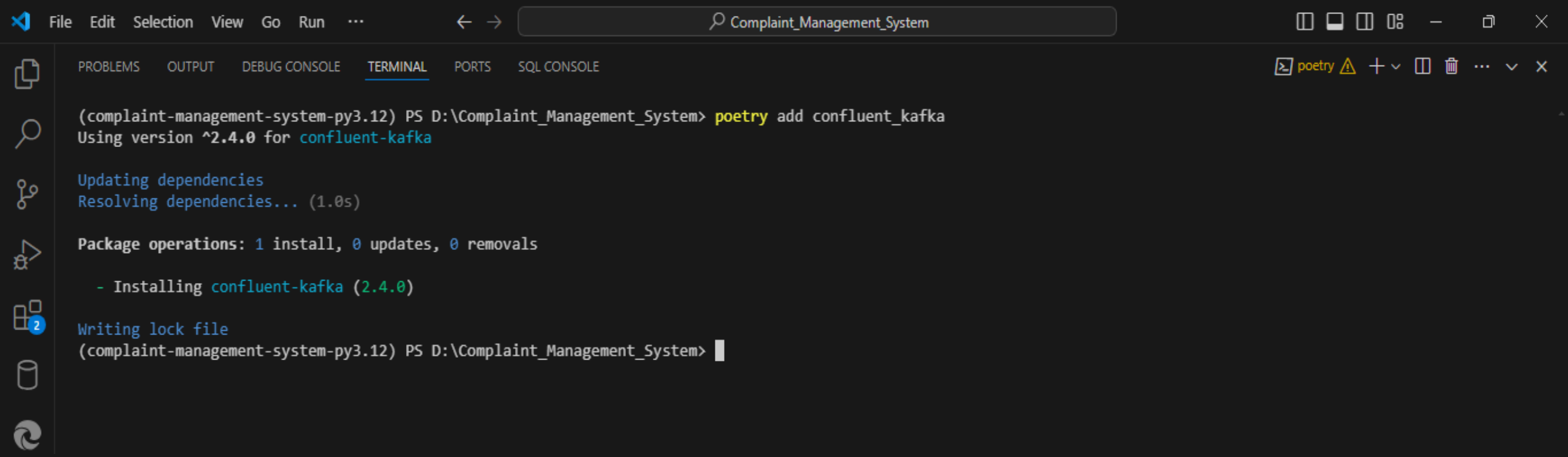
This PC › Local Disk (D:) › Complaint_Management_System

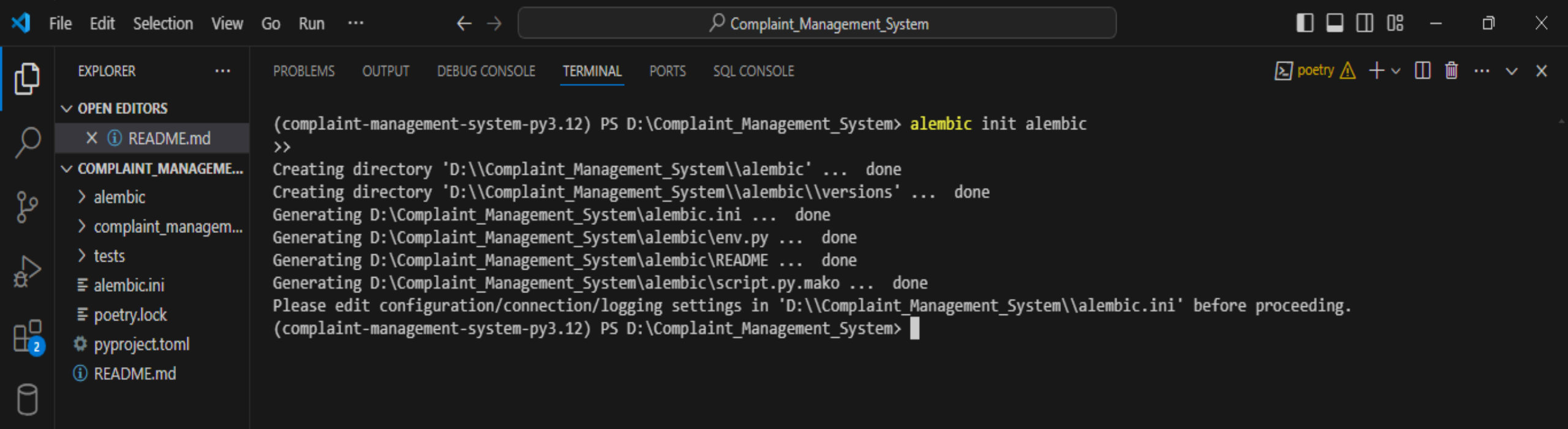
Search Complaint_Management_System

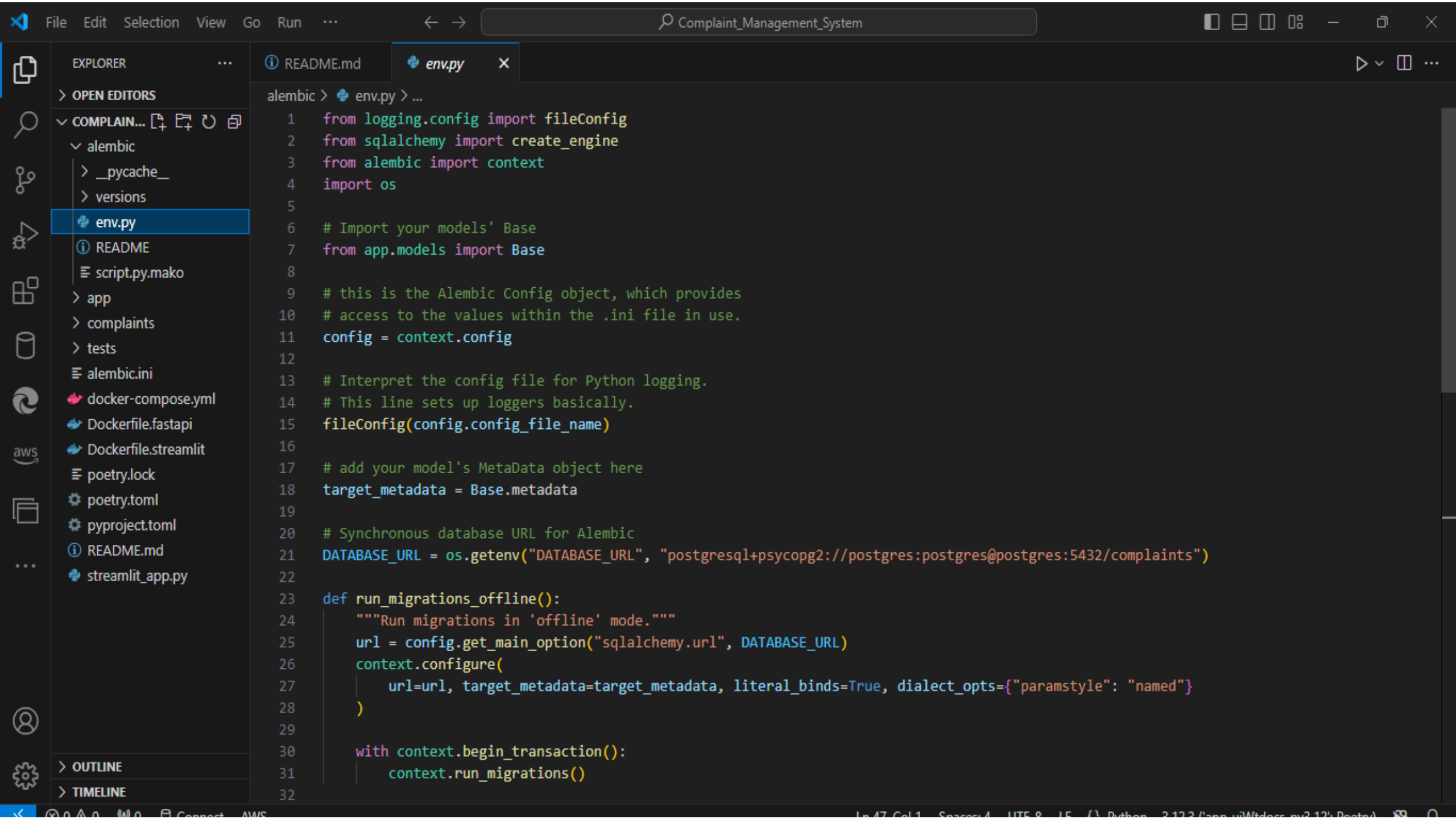
Name	Date modified	Type	Size
complaint_management_system	23-Jun-24 12:50 PM	File folder	
tests	23-Jun-24 12:50 PM	File folder	
poetry.lock	23-Jun-24 1:01 PM	LOCK File	236 KB
pyproject.toml	23-Jun-24 1:01 PM	Toml Source File	1 KB
README.md	23-Jun-24 12:50 PM	Markdown Source...	0 KB

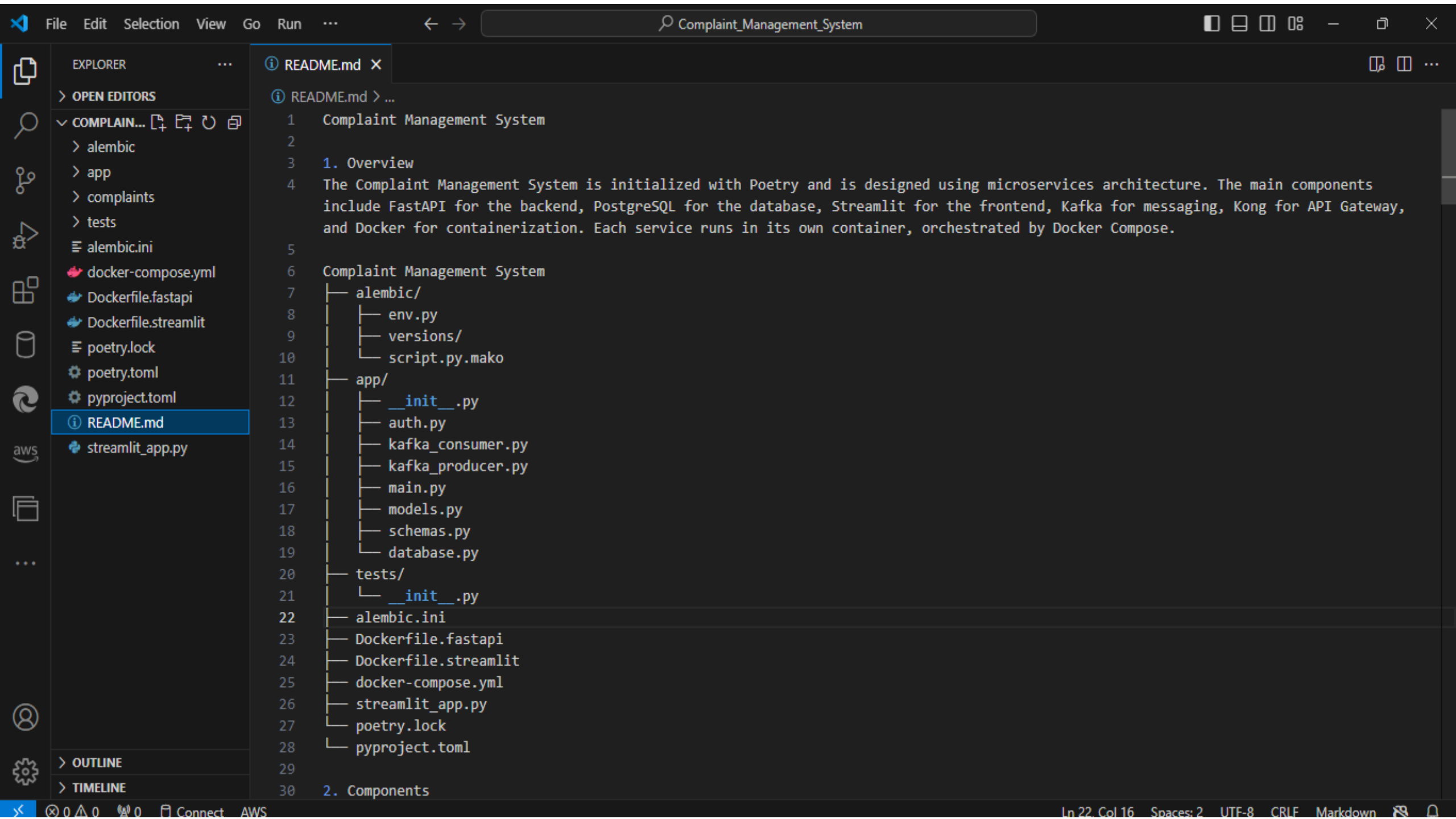
```
(complaint-management-system-py3.12) D:\Complaint_Management_System>poetry add fastapi uvicorn[standard] sqlalchemy asyncpg alembic pydantic kafka-python psycopg2-binar
y streamlit cryptography
Using version ^0.111.0 for fastapi
Using version ^0.30.1 for uvicorn
Using version ^2.0.31 for sqlalchemy
Using version ^0.29.0 for asyncpg
Using version ^1.13.1 for alembic
Using version ^2.7.4 for pydantic
Using version ^2.0.2 for kafka-python
Using version ^2.9.9 for psycopg2-binary
Using version ^1.36.0 for streamlit
Using version ^42.0.8 for cryptography

Updating dependencies
Resolving dependencies... (16.5s)

Package operations: 76 installs, 0 updates, 0 removals

  - Installing attrs (23.2.0)
  - Installing mdurl (0.1.2)
  - Installing rpds-py (0.18.1)
  - Installing colorama (0.4.6)
  - Installing idna (3.7)
  - Installing markdown-it-py (3.0.0)
  - Installing pygments (2.18.0)
  - Installing referencing (0.35.1)
  - Installing six (1.16.0)
  - Installing sniffio (1.3.1)
  - Installing anyio (4.4.0)
  - Installing certifi (2024.6.2)
  - Installing click (8.1.7)
  - Installing h11 (0.14.0)
  - Installing jsonschema-specifications (2023.12.1)
  - Installing markupsafe (2.1.5)
  - Installing numpy (2.0.0)
  - Installing python-dateutil (2.9.0.post0)
  - Installing pytz (2024.1)
  - Installing rich (13.7.1)
  - Installing shellingham (1.5.4)
  - Installing smmap (5.0.1)
  - Installing typing-extensions (4.12.2)
  - Installing tzdata (2024.1)
```

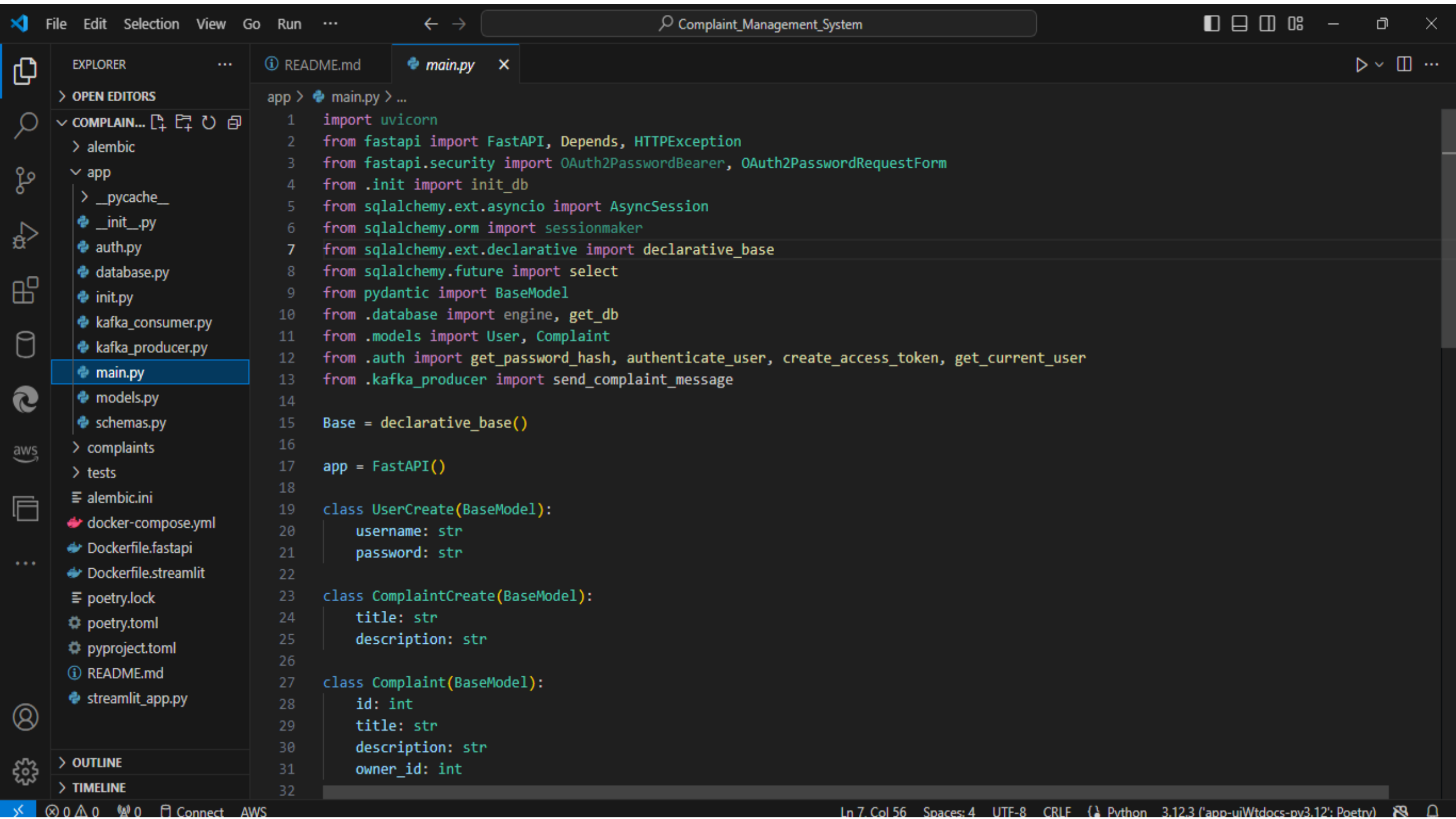

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
pyproject.toml
OPEN EDITORS
COMPLAIN...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
pyproject.toml
[tool.poetry]
name = "app"
version = "0.1.0"
description = ""
authors = ["TAJ AHMED <141453003+AuditorTaj@users.noreply.github.com>"]
readme = "README.md"
[tool.poetry.dependencies]
python = "^3.12"
fastapi = "^0.111.0"
uvicorn = {extras = ["standard"], version = "^0.30.1"}
sqlalchemy = "^2.0.31"
asyncpg = "^0.29.0"
alembic = "^1.13.1"
pydantic = "^2.7.4"
kafka-python = "^2.0.2"
psycopg2-binary = "^2.9.9"
streamlit = "^1.36.0"
cryptography = "^42.0.8"
confluent-kafka = "^2.4.0"
pandas = "^2.2.2"
databases = "^0.9.0"
passlib = "^1.7.4"
pyjwt = "^2.8.0"
aiosqlite = "^0.20.0"
typing-extensions = "^4.12.2"
kong = "^0.2.1"
postgres = "^4.0"
neon = "^0.1.2"
python-jose = "^3.3.0"
requests = "^2.32.3"
aiohttp = "^3.9.5"

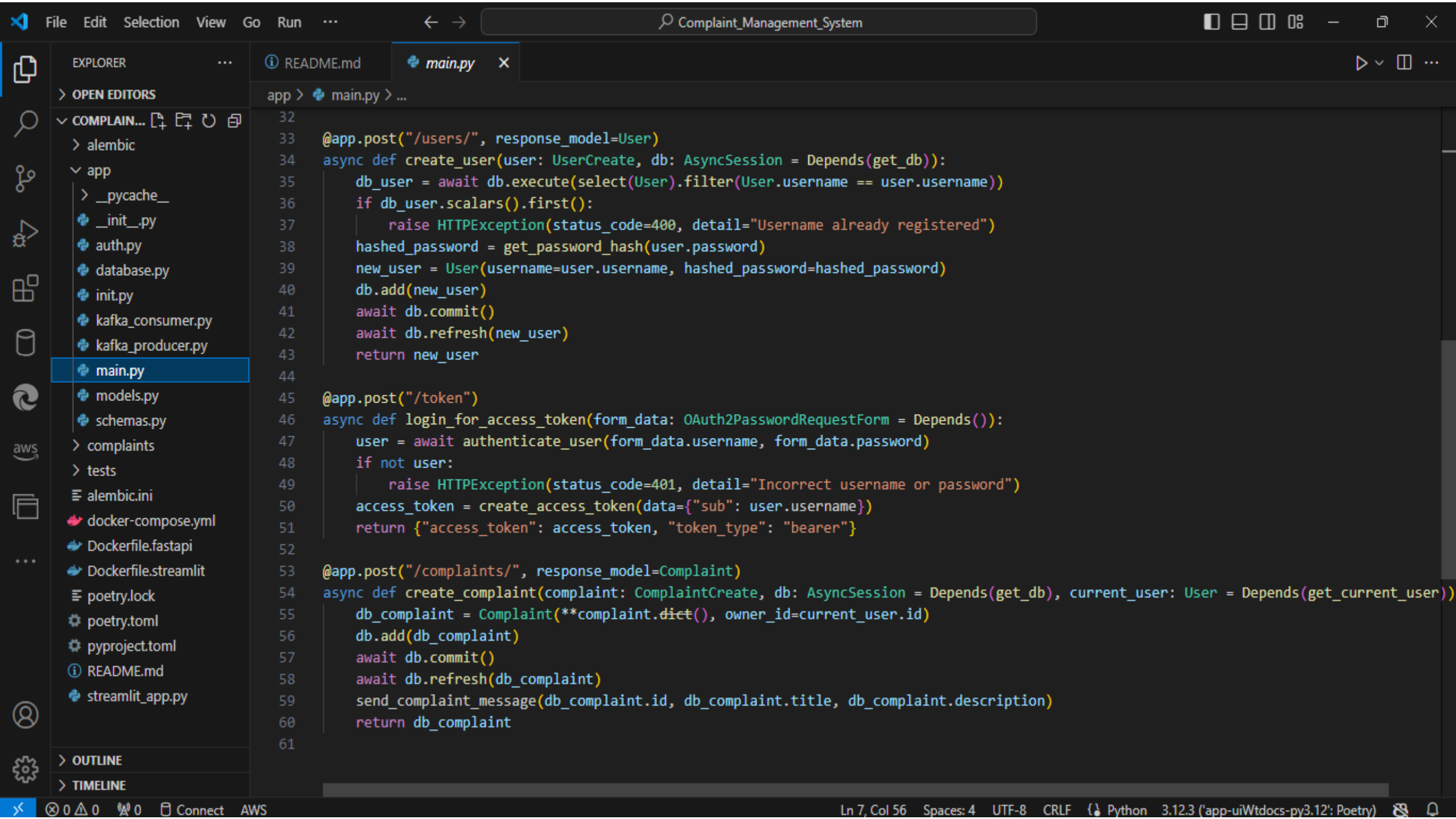

File Edit Selection View Go Run ···
Complaint_Management_System
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE
poetry
(complaint-management-system-py3.12) PS D:\Complaint_Management_System> poetry add confluent_kafka
Using version ^2.4.0 for confluent-kafka
Updating dependencies
Resolving dependencies... (1.0s)
Package operations: 1 install, 0 updates, 0 removals
- Installing confluent-kafka (2.4.0)
Writing lock file
(complaint-management-system-py3.12) PS D:\Complaint_Management_System>

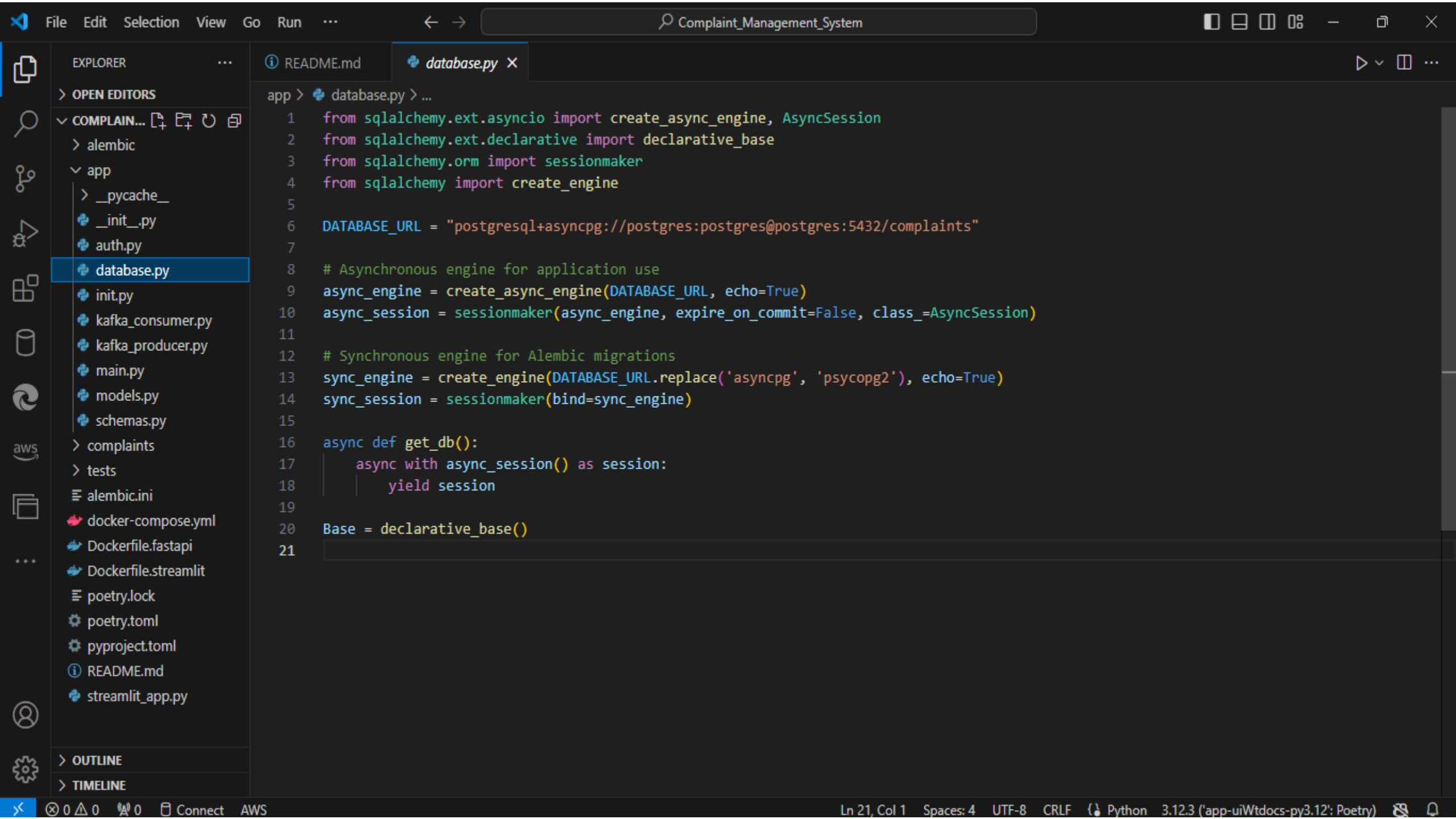

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
OPEN EDITORS
README.md
COMPLAINT_MANAGEME...
alembic
complaint_managem...
tests
alembic.ini
poetry.lock
pyproject.toml
README.md
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE
poetry
(complaint-management-system-py3.12) PS D:\Complaint_Management_System> alembic init alembic
>>
Creating directory 'D:\\Complaint_Management_System\\alembic' ... done
Creating directory 'D:\\Complaint_Management_System\\alembic\\versions' ... done
Generating D:\Complaint_Management_System\alembic.ini ... done
Generating D:\Complaint_Management_System\alembic\env.py ... done
Generating D:\Complaint_Management_System\alembic\README ... done
Generating D:\Complaint_Management_System\alembic\script.py.mako ... done
Please edit configuration/connection/logging settings in 'D:\\Complaint_Management_System\\alembic.ini' before proceeding.
(complaint-management-system-py3.12) PS D:\Complaint_Management_System>

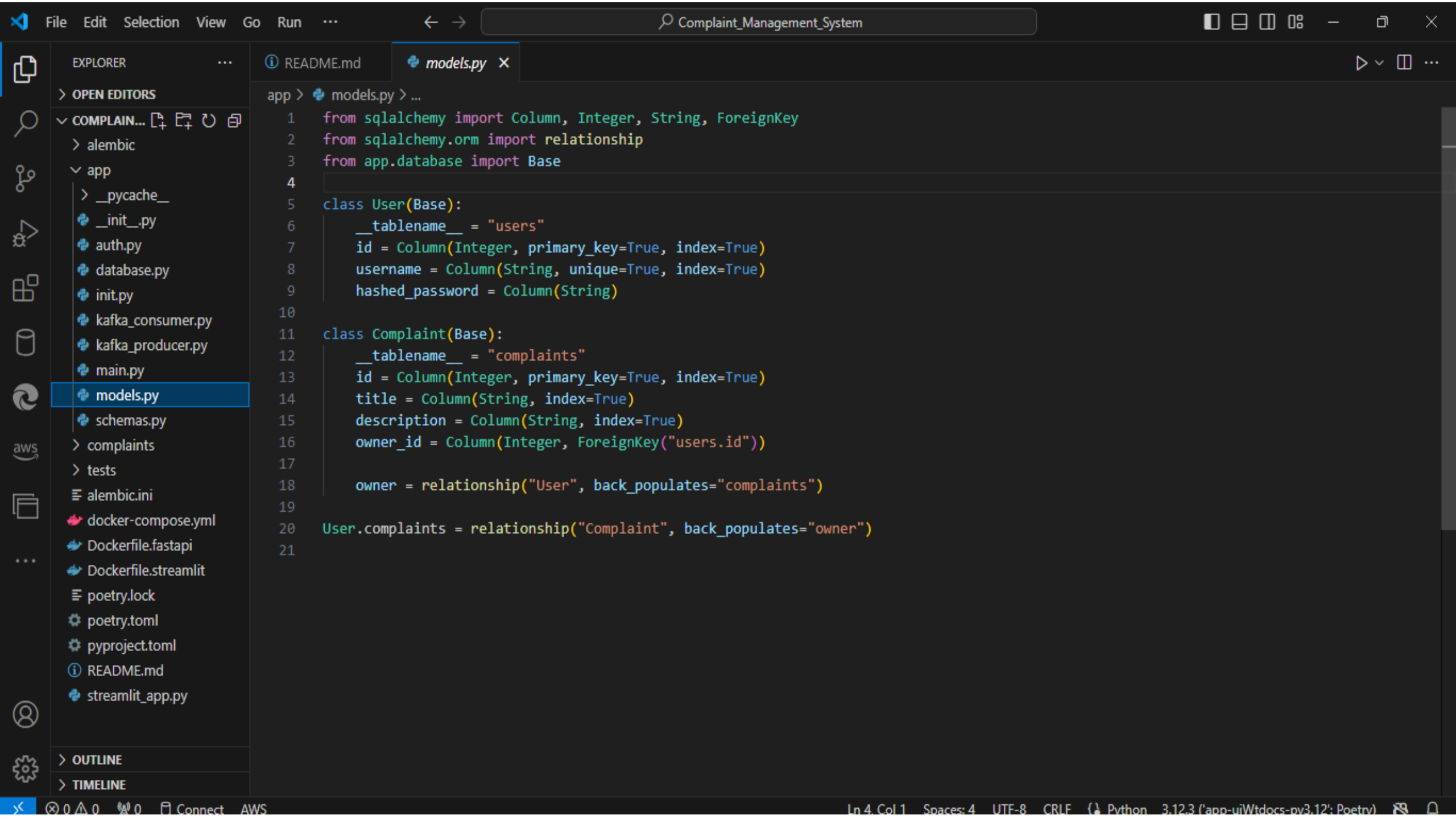

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
env.py
alembic > env.py > ...
OPEN EDITORS
COMPLAIN...
alembic
__pycache__
versions
env.py
README
script.py.mako
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
from logging.config import fileConfig
from sqlalchemy import create_engine
from alembic import context
import os
# Import your models' Base
from app.models import Base
# this is the Alembic Config object, which provides
# access to the values within the .ini file in use.
config = context.config
# Interpret the config file for Python logging.
# This line sets up loggers basically.
fileConfig(config.config_file_name)
# add your model's MetaData object here
target_metadata = Base.metadata
# Synchronous database URL for Alembic
DATABASE_URL = os.getenv("DATABASE_URL", "postgresql+psycopg2://postgres:postgres@postgres:5432/complaints")
def run_migrations_offline():
"""Run migrations in 'offline' mode."""
url = config.get_main_option("sqlalchemy.url", DATABASE_URL)
context.configure(
url=url, target_metadata=target_metadata, literal_binds=True, dialect_opts={"paramstyle": "named"}
)
with context.begin_transaction():
context.run_migrations()

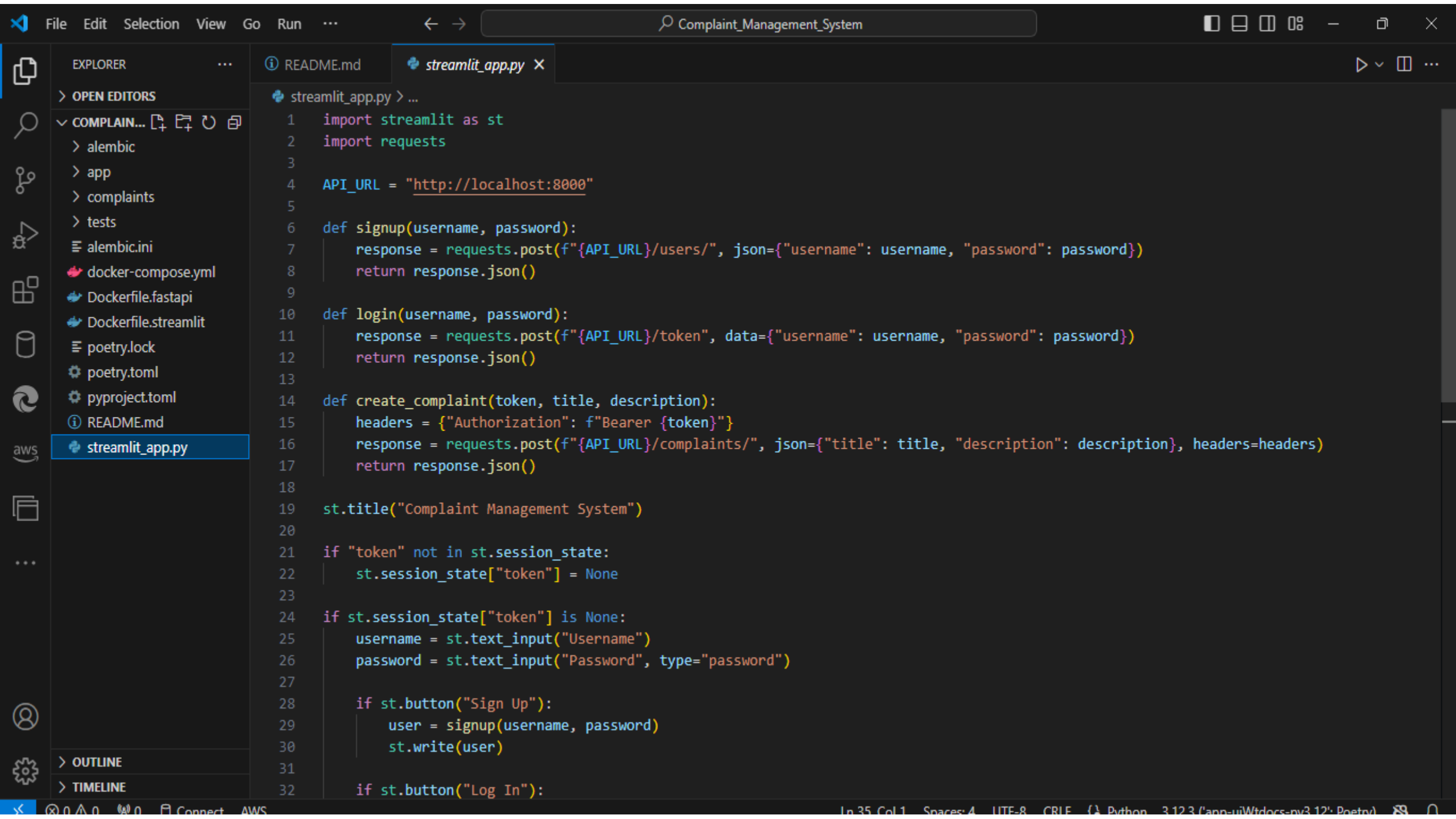

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
OPEN EDITORS
COMPLAIN...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
README.md > ...
Complaint Management System
1. Overview
The Complaint Management System is initialized with Poetry and is designed using microservices architecture. The main components include FastAPI for the backend, PostgreSQL for the database, Streamlit for the frontend, Kafka for messaging, Kong for API Gateway, and Docker for containerization. Each service runs in its own container, orchestrated by Docker Compose.
Complaint Management System
├── alembic/
│ ├── env.py
│ ├── versions/
│ └── script.py.mako
├── app/
│ ├── __init__.py
│ ├── auth.py
│ ├── kafka_consumer.py
│ ├── kafka_producer.py
│ ├── main.py
│ ├── models.py
│ ├── schemas.py
│ └── database.py
├── tests/
│ └── __init__.py
├── alembic.ini
├── Dockerfile.fastapi
├── Dockerfile.streamlit
├── docker-compose.yml
├── streamlit_app.py
└── poetry.lock
└── pyproject.toml
2. Components
Connect AWS
Ln 22, Col 16 Spaces: 2 UTF-8 CRLF Markdown

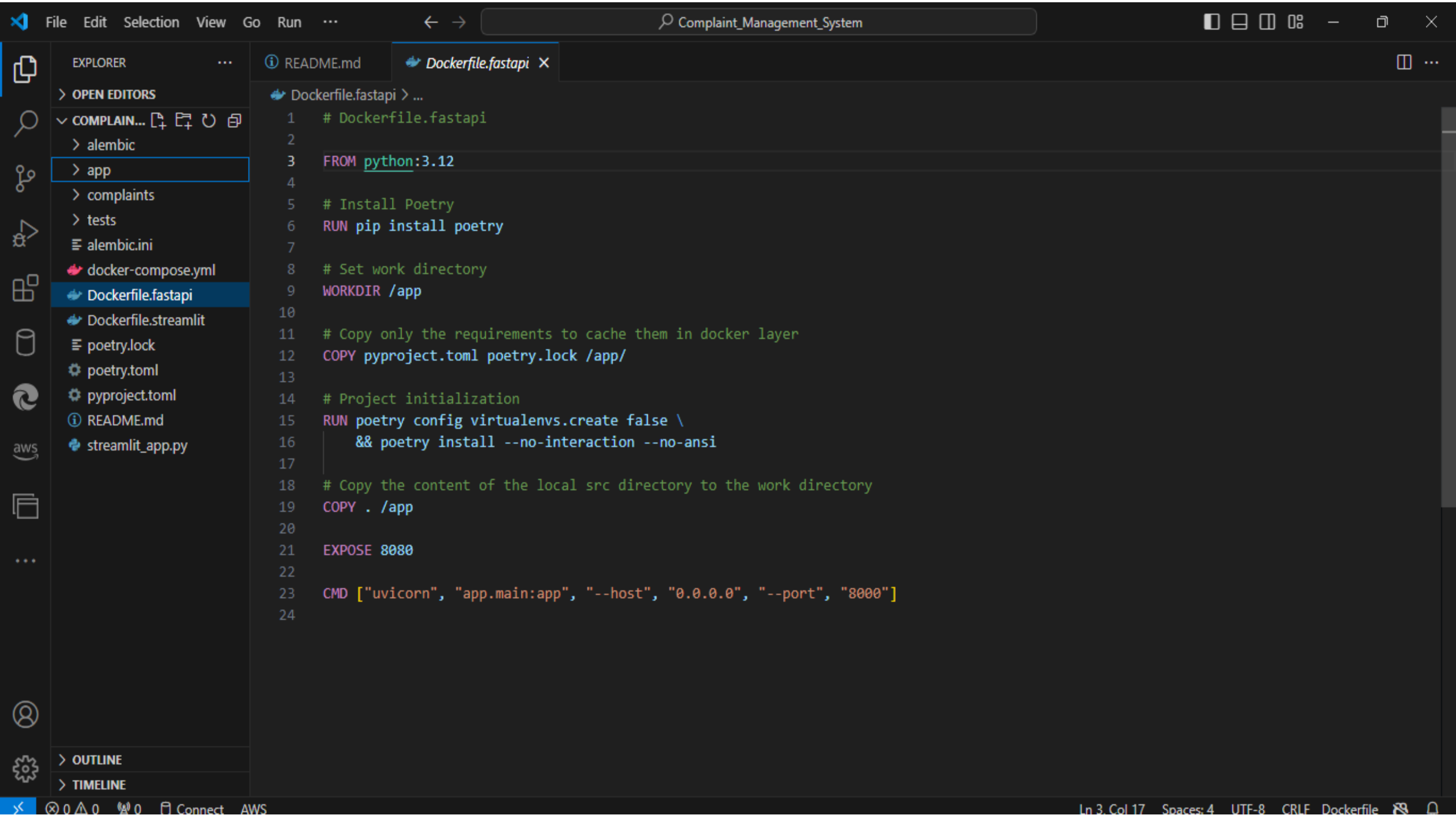

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
main.py
app > main.py > ...
OPEN EDITORS
COMPLAIN...
alembic
app
__pycache__
__init__.py
auth.py
database.py
init.py
kafka_consumer.py
kafka_producer.py
main.py
models.py
schemas.py
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
import uvicorn
from fastapi import FastAPI, Depends, HTTPException
from fastapi.security import OAuth2PasswordBearer, OAuth2PasswordRequestForm
from .init import init_db
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.future import select
from pydantic import BaseModel
from .database import engine, get_db
from .models import User, Complaint
from .auth import get_password_hash, authenticate_user, create_access_token, get_current_user
from .kafka_producer import send_complaint_message
Base = declarative_base()
app = FastAPI()
class UserCreate(BaseModel):
username: str
password: str
class ComplaintCreate(BaseModel):
title: str
description: str
class Complaint(BaseModel):
id: int
title: str
description: str
owner_id: int
Connect AWS
Ln 7, Col 56 Spaces: 4 UTF-8 CRLF Python 3.12.3 ('app-uiWtdocs-py3.12': Poetry)

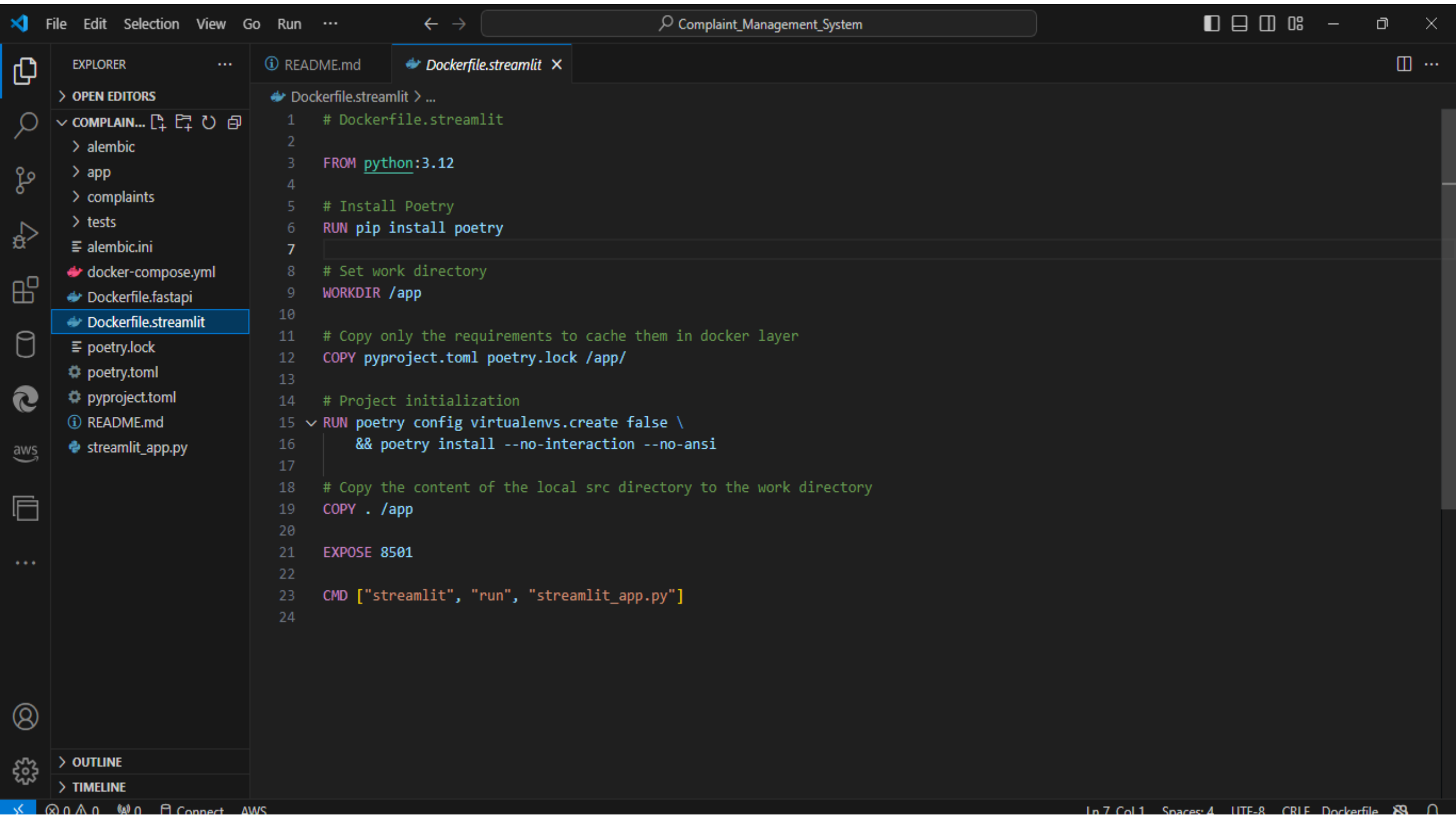

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
main.py
app > main.py > ...
OPEN EDITORS
COMPLAIN...
alembic
app
__pycache__
__init__.py
auth.py
database.py
init.py
kafka_consumer.py
kafka_producer.py
main.py
models.py
schemas.py
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
@app.post("/users/", response_model=User)
async def create_user(user: UserCreate, db: AsyncSession = Depends(get_db)):
db_user = await db.execute(select(User).filter(User.username == user.username))
if db_user.scalars().first():
raise HTTPException(status_code=400, detail="Username already registered")
hashed_password = get_password_hash(user.password)
new_user = User(username=user.username, hashed_password=hashed_password)
db.add(new_user)
await db.commit()
await db.refresh(new_user)
return new_user
@app.post("/token")
async def login_for_access_token(form_data: OAuth2PasswordRequestForm = Depends()):
user = await authenticate_user(form_data.username, form_data.password)
if not user:
raise HTTPException(status_code=401, detail="Incorrect username or password")
access_token = create_access_token(data={"sub": user.username})
return {"access_token": access_token, "token_type": "bearer"}
@app.post("/complaints/", response_model=Complaint)
async def create_complaint(complaint: ComplaintCreate, db: AsyncSession = Depends(get_db), current_user: User = Depends(get_current_user))
db_complaint = Complaint(**complaint.dict(), owner_id=current_user.id)
db.add(db_complaint)
await db.commit()
await db.refresh(db_complaint)
send_complaint_message(db_complaint.id, db_complaint.title, db_complaint.description)
return db_complaint
Ln 7, Col 56 Spaces: 4 UTF-8 CRLF Python 3.12.3 ('app-uiWtdocs-py3.12': Poetry)
Connect AWS

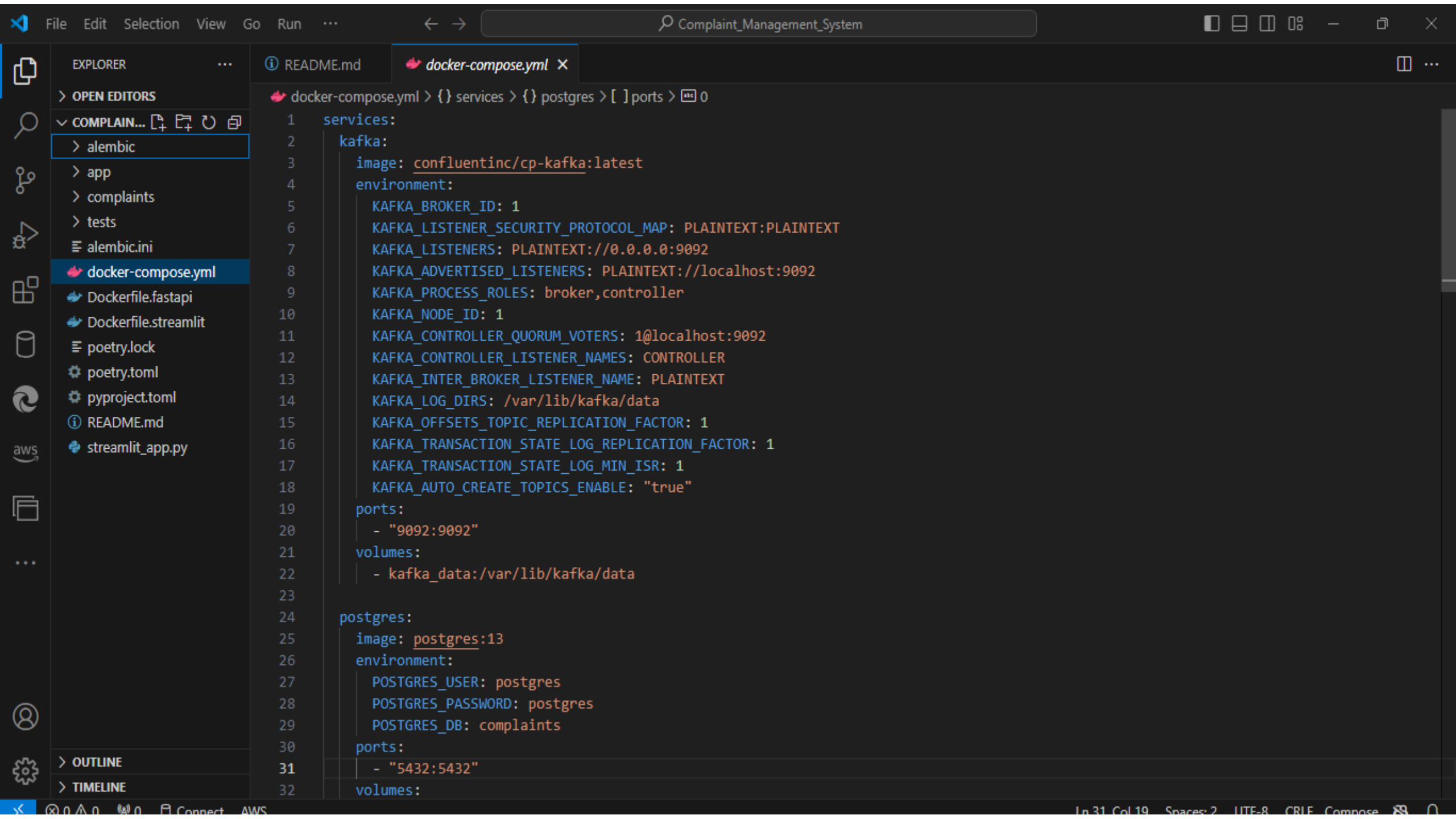

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
database.py
app > database.py > ...
OPEN EDITORS
COMPLAIN...
alembic
app
__pycache__
__init__.py
auth.py
database.py
init.py
kafka_consumer.py
kafka_producer.py
main.py
models.py
schemas.py
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
1 from sqlalchemy.ext.asyncio import create_async_engine, AsyncSession
2 from sqlalchemy.ext.declarative import declarative_base
3 from sqlalchemy.orm import sessionmaker
4 from sqlalchemy import create_engine
5
6 DATABASE_URL = "postgresql+asyncpg://postgres:postgres@postgres:5432/complaints"
7
8 # Asynchronous engine for application use
9 async_engine = create_async_engine(DATABASE_URL, echo=True)
10 async_session = sessionmaker(async_engine, expire_on_commit=False, class_=AsyncSession)
11
12 # Synchronous engine for Alembic migrations
13 sync_engine = create_engine(DATABASE_URL.replace('asyncpg', 'psycopg2'), echo=True)
14 sync_session = sessionmaker(bind=sync_engine)
15
16 async def get_db():
17 async with async_session() as session:
18 yield session
19
20 Base = declarative_base()
21
Ln 21, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.3 ('app-uiWtdocs-py3.12': Poetry)

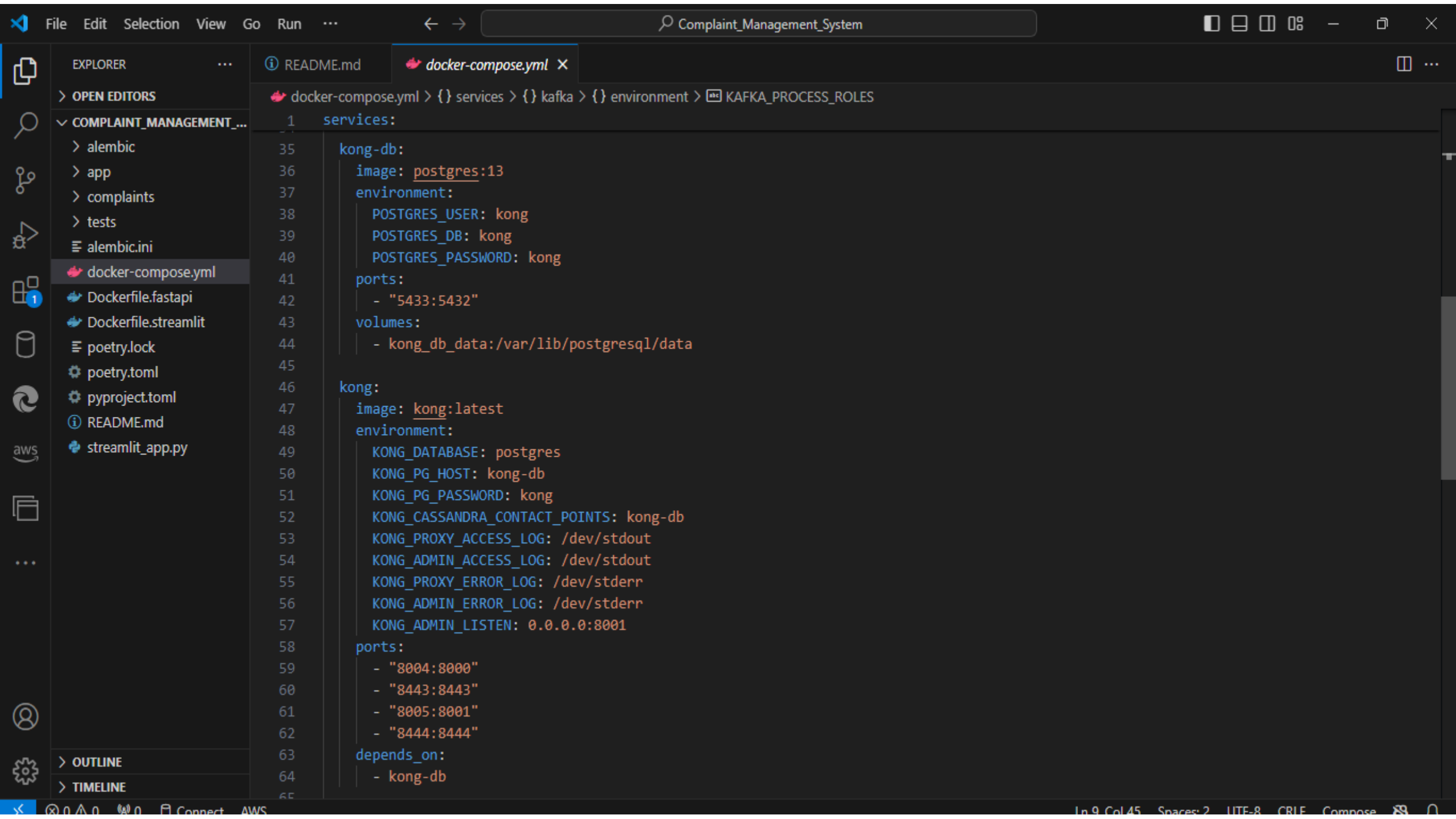

File Edit Selection View Go Run
Complaint_Management_System
EXPLORER
README.md
models.py
app > models.py > ...
OPEN EDITORS
COMPLAIN...
alembic
app
__pycache__
__init__.py
auth.py
database.py
init.py
kafka_consumer.py
kafka_producer.py
main.py
models.py
schemas.py
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
1 from sqlalchemy import Column, Integer, String, ForeignKey
2 from sqlalchemy.orm import relationship
3 from app.database import Base
4
5 class User(Base):
6 __tablename__ = "users"
7 id = Column(Integer, primary_key=True, index=True)
8 username = Column(String, unique=True, index=True)
9 hashed_password = Column(String)
10
11 class Complaint(Base):
12 __tablename__ = "complaints"
13 id = Column(Integer, primary_key=True, index=True)
14 title = Column(String, index=True)
15 description = Column(String, index=True)
16 owner_id = Column(Integer, ForeignKey("users.id"))
17
18 owner = relationship("User", back_populates="complaints")
19
20 User.complaints = relationship("Complaint", back_populates="owner")
21
Connect AWS
Ln 4, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.3 ('app-uiWtdocs-py3.12': Poetry)

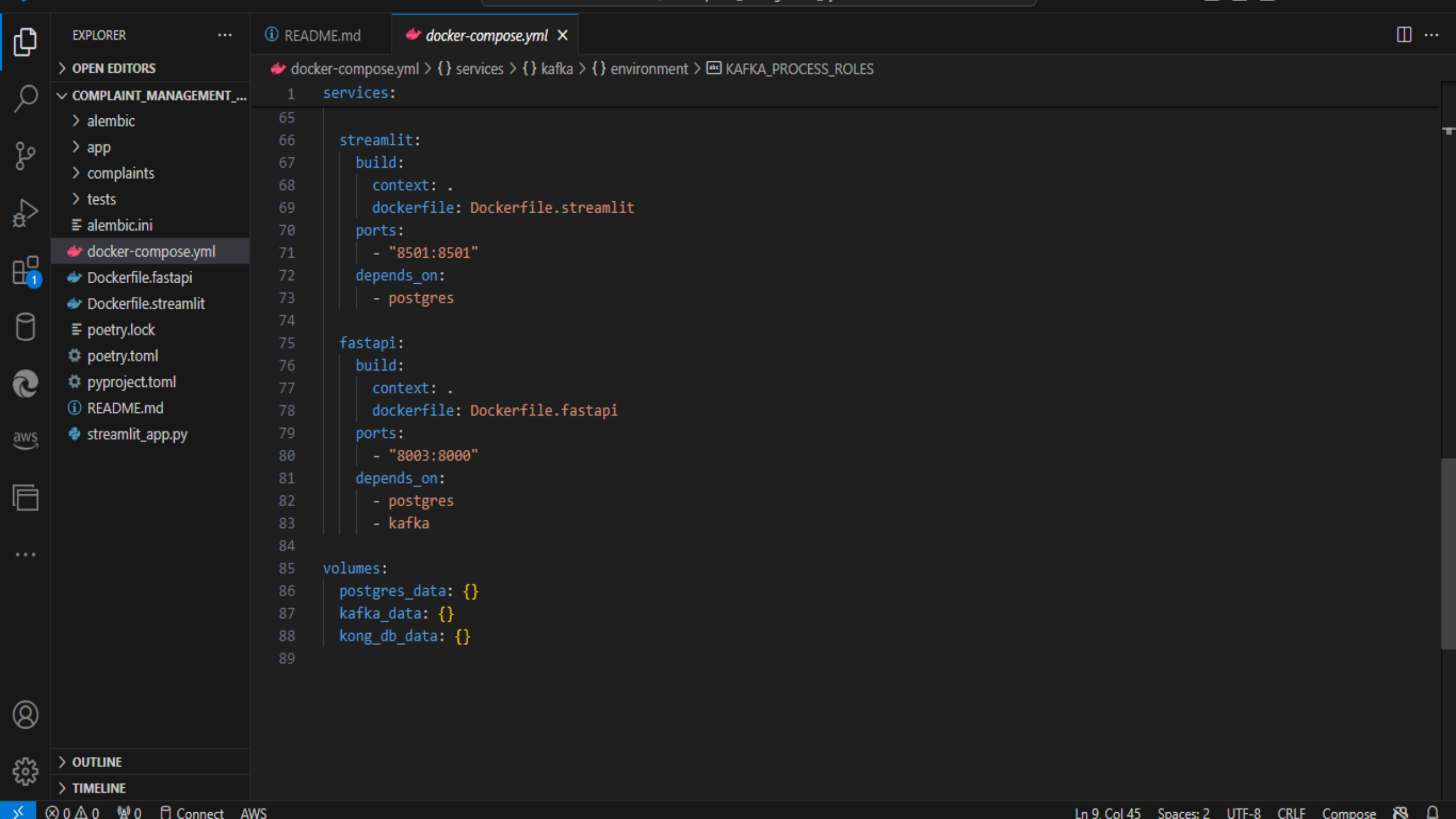

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
streamlit_app.py
OPEN EDITORS
COMPLAIN...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
streamlit_app.py > ...
1 import streamlit as st
2 import requests
3
4 API_URL = "http://localhost:8000"
5
6 def signup(username, password):
7 response = requests.post(f"{API_URL}/users/", json={"username": username, "password": password})
8 return response.json()
9
10 def login(username, password):
11 response = requests.post(f"{API_URL}/token", data={"username": username, "password": password})
12 return response.json()
13
14 def create_complaint(token, title, description):
15 headers = {"Authorization": f"Bearer {token}"}
16 response = requests.post(f"{API_URL}/complaints/", json={"title": title, "description": description}, headers=headers)
17 return response.json()
18
19 st.title("Complaint Management System")
20
21 if "token" not in st.session_state:
22 st.session_state["token"] = None
23
24 if st.session_state["token"] is None:
25 username = st.text_input("Username")
26 password = st.text_input("Password", type="password")
27
28 if st.button("Sign Up"):
29 user = signup(username, password)
30 st.write(user)
31
32 if st.button("Log In"):
Connect AWS
Ln 35, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.3 ('app-uiWtdocs-py3.12': Poetry)

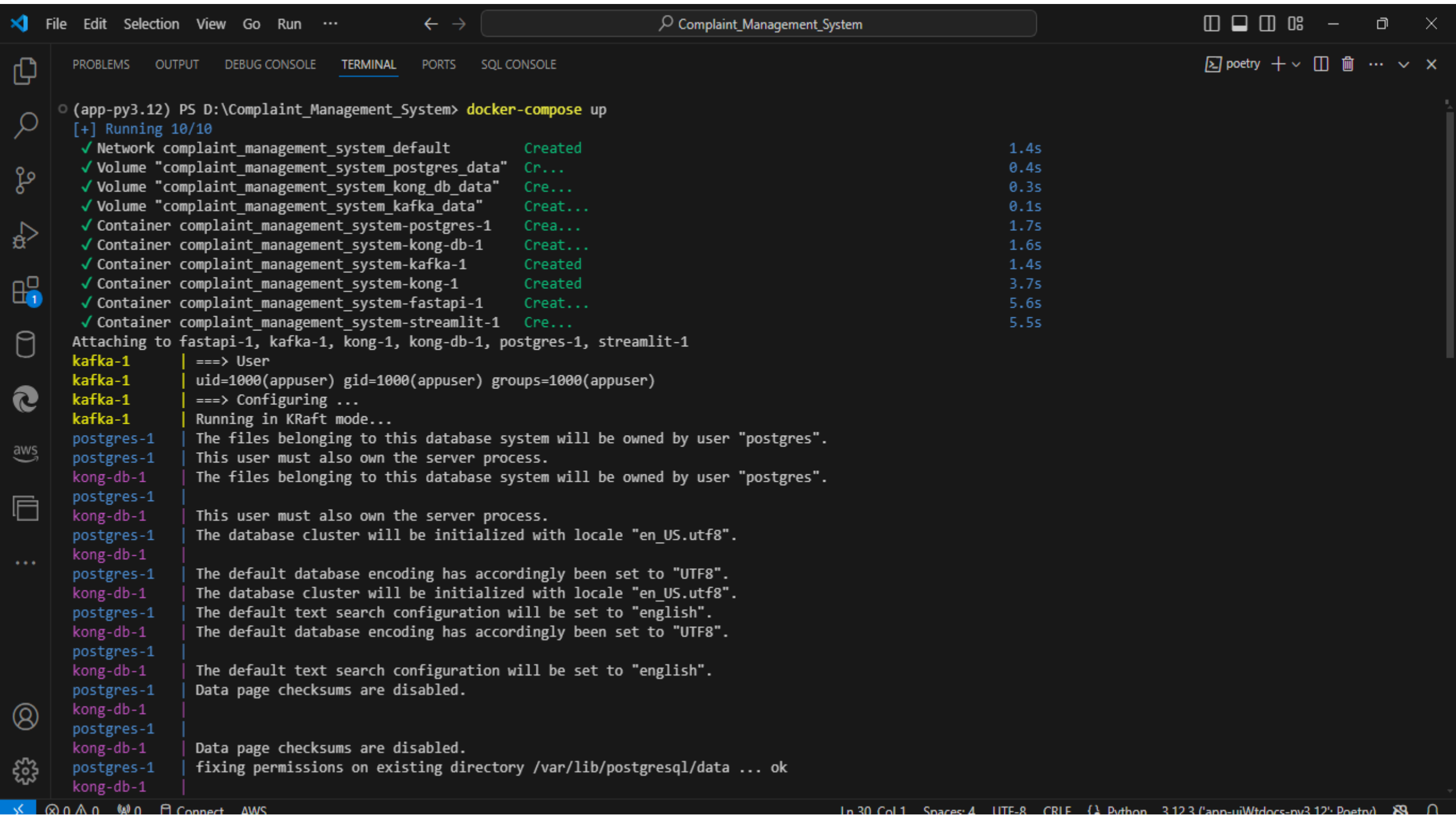

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
Dockerfile.fastapi
Dockerfile.fastapi > ...
OPEN EDITORS
COMPLAIN...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
1 # Dockerfile.fastapi
2
3 FROM python:3.12
4
5 # Install Poetry
6 RUN pip install poetry
7
8 # Set work directory
9 WORKDIR /app
10
11 # Copy only the requirements to cache them in docker layer
12 COPY pyproject.toml poetry.lock /app/
13
14 # Project initialization
15 RUN poetry config virtualenvs.create false \
16 && poetry install --no-interaction --no-ansi
17
18 # Copy the content of the local src directory to the work directory
19 COPY . /app
20
21 EXPOSE 8080
22
23 CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000"]
24
0 0 0 Connect AWS
Ln 3, Col 17 Spaces: 4 UTF-8 CRLF Dockerfile

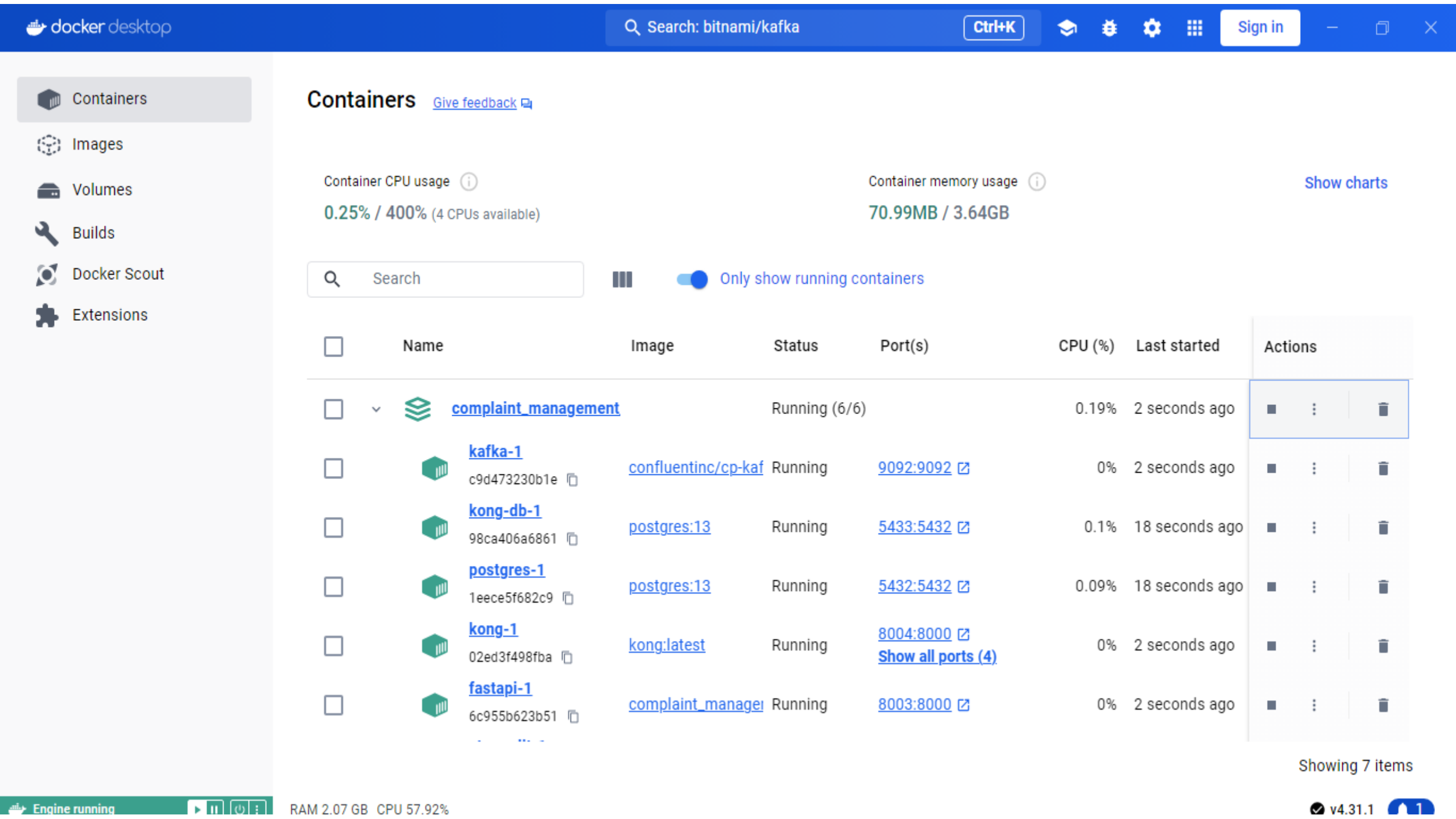

File Edit Selection View Go Run
Complaint_Management_System
EXPLORER
README.md
Dockerfile.streamlit
OPEN EDITORS
COMPLAIN...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
Dockerfile.streamlit > ...
1 # Dockerfile.streamlit
2
3 FROM python:3.12
4
5 # Install Poetry
6 RUN pip install poetry
7
8 # Set work directory
9 WORKDIR /app
10
11 # Copy only the requirements to cache them in docker layer
12 COPY pyproject.toml poetry.lock /app/
13
14 # Project initialization
15 RUN poetry config virtualenvs.create false \
16 && poetry install --no-interaction --no-ansi
17
18 # Copy the content of the local src directory to the work directory
19 COPY . /app
20
21 EXPOSE 8501
22
23 CMD ["streamlit", "run", "streamlit_app.py"]
24
Connect AWS
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF Dockerfile

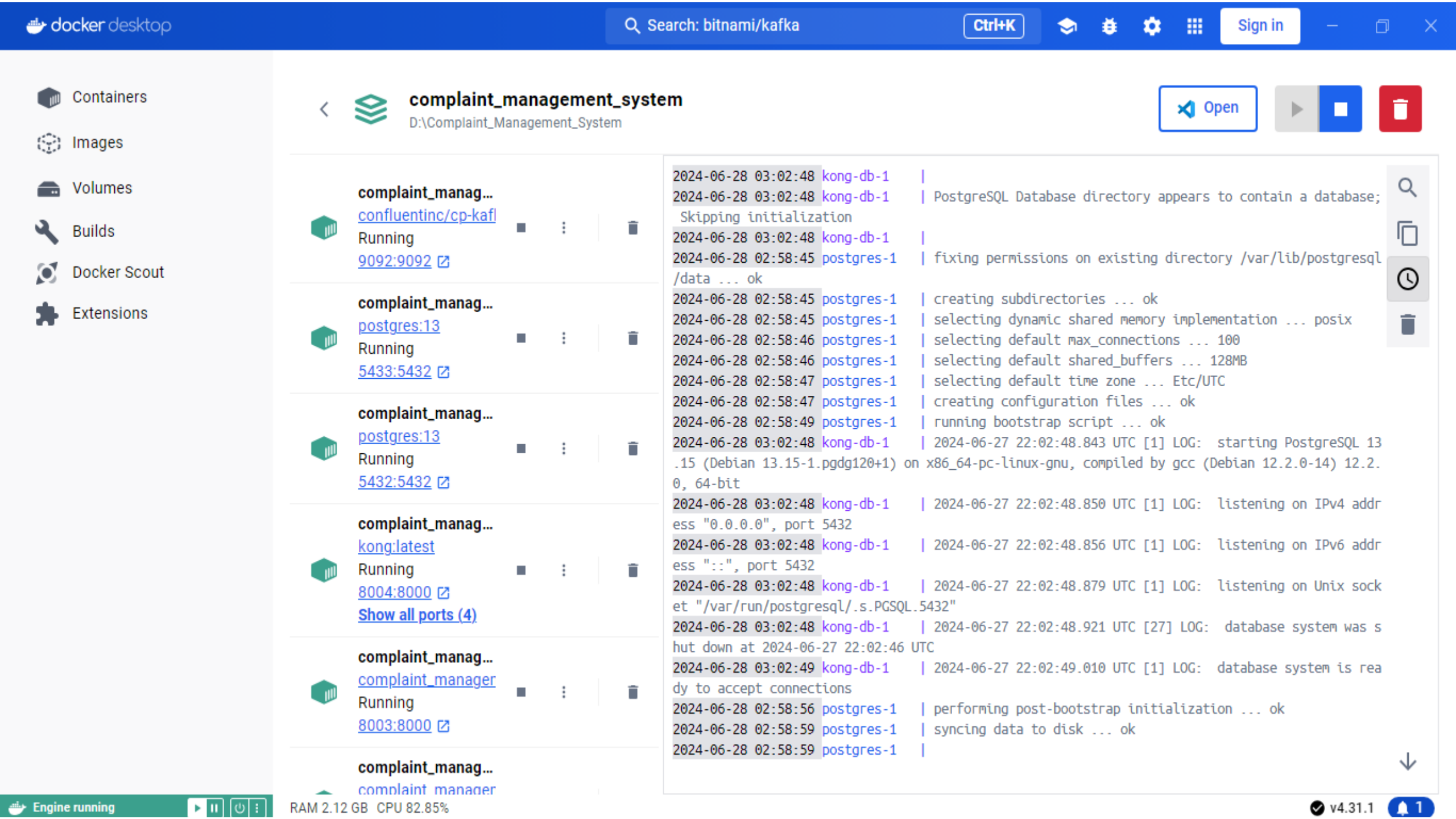

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
docker-compose.yml
docker-compose.yml > { } services > { } postgres > [ ] ports > 0
OPEN EDITORS
COMPLAIN...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
services:
kafka:
image: confluentinc/cp-kafka:latest
environment:
KAFKA_BROKER_ID: 1
KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT
KAFKA_LISTENERS: PLAINTEXT://0.0.0.0:9092
KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
KAFKA_PROCESS_ROLES: broker,controller
KAFKA_NODE_ID: 1
KAFKA_CONTROLLER_QUORUM_VOTERS: 1@localhost:9092
KAFKA_CONTROLLER_LISTENER_NAMES: CONTROLLER
KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
KAFKA_LOG_DIRS: /var/lib/kafka/data
KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1
KAFKA_TRANSACTION_STATE_LOG_REPLICATION_FACTOR: 1
KAFKA_TRANSACTION_STATE_LOG_MIN_ISR: 1
KAFKA_AUTO_CREATE_TOPICS_ENABLE: "true"
ports:
- "9092:9092"
volumes:
- kafka_data:/var/lib/kafka/data
postgres:
image: postgres:13
environment:
POSTGRES_USER: postgres
POSTGRES_PASSWORD: postgres
POSTGRES_DB: complaints
ports:
- "5432:5432"
volumes:
Connect AWS
Ln 31, Col 19 Spaces: 2 UTF-8 CRLF Compose

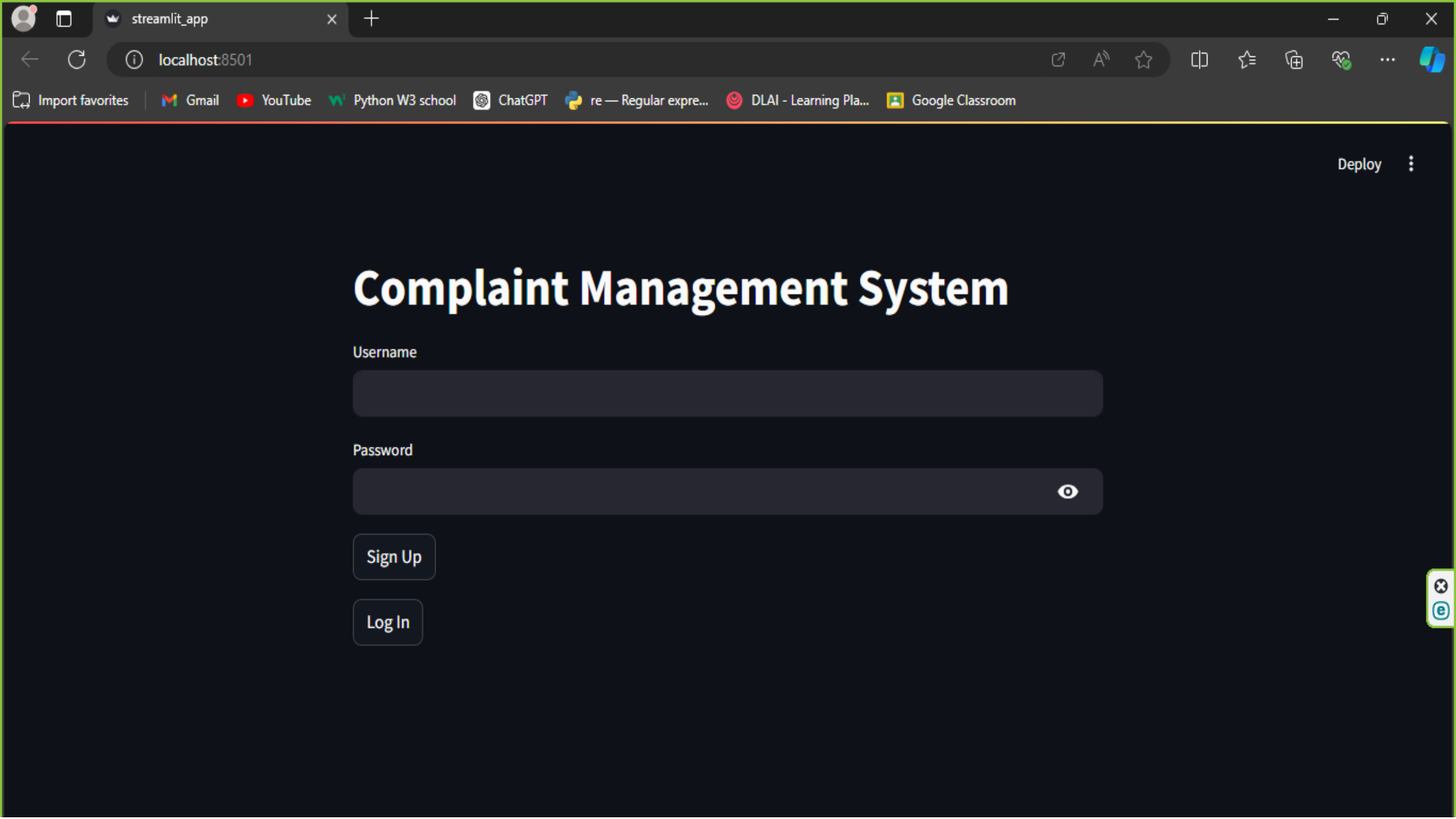

File Edit Selection View Go Run ···
Complaint_Management_System
EXPLORER
README.md
docker-compose.yml
docker-compose.yml > { } services > { } kafka > { } environment > KAFKA_PROCESS_ROLES
OPEN EDITORS
COMPLAINT_MANAGEMENT_...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
1 services:
35 kong-db:
36 image: postgres:13
37 environment:
38 POSTGRES_USER: kong
39 POSTGRES_DB: kong
40 POSTGRES_PASSWORD: kong
41 ports:
42 - "5433:5432"
43 volumes:
44 - kong_db_data:/var/lib/postgresql/data
45
46 kong:
47 image: kong:latest
48 environment:
49 KONG_DATABASE: postgres
50 KONG_PG_HOST: kong-db
51 KONG_PG_PASSWORD: kong
52 KONG_CASSANDRA_CONTACT_POINTS: kong-db
53 KONG_PROXY_ACCESS_LOG: /dev/stdout
54 KONG_ADMIN_ACCESS_LOG: /dev/stdout
55 KONG_PROXY_ERROR_LOG: /dev/stderr
56 KONG_ADMIN_ERROR_LOG: /dev/stderr
57 KONG_ADMIN_LISTEN: 0.0.0.0:8001
58 ports:
59 - "8004:8000"
60 - "8443:8443"
61 - "8005:8001"
62 - "8444:8444"
63 depends_on:
64 - kong-db
Connect AWS
Ln 9, Col 45 Spaces: 2 UTF-8 CRLF Compose


EXPLORER
README.md
docker-compose.yml
docker-compose.yml > { } services > { } kafka > { } environment > KAFKA_PROCESS_ROLES
OPEN EDITORS
COMPLAINT_MANAGEMENT_...
alembic
app
complaints
tests
alembic.ini
docker-compose.yml
Dockerfile.fastapi
Dockerfile.streamlit
poetry.lock
poetry.toml
pyproject.toml
README.md
streamlit_app.py
OUTLINE
TIMELINE
1 services:
65
66 streamlit:
67 build:
68 context: .
69 dockerfile: Dockerfile.streamlit
70 ports:
71 - "8501:8501"
72 depends_on:
73 - postgres
74
75 fastapi:
76 build:
77 context: .
78 dockerfile: Dockerfile.fastapi
79 ports:
80 - "8003:8000"
81 depends_on:
82 - postgres
83 - kafka
84
85 volumes:
86 postgres_data: {}
87 kafka_data: {}
88 kong_db_data: {}
89
Connect
AWS
Ln 9, Col 45
Spaces: 2
UTF-8
CRLF
Compose


File Edit Selection View Go Run ···
Complaint_Management_System
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE
poetry
(app-py3.12) PS D:\Complaint_Management_System> docker-compose up
[+] Running 10/10
✓ Network complaint_management_system_default Created 1.4s
✓ Volume "complaint_management_system_postgres_data" Cr... 0.4s
✓ Volume "complaint_management_system_kong_db_data" Cre... 0.3s
✓ Volume "complaint_management_system_kafka_data" Creat... 0.1s
✓ Container complaint_management_system-postgres-1 Crea... 1.7s
✓ Container complaint_management_system-kong-db-1 Creat... 1.6s
✓ Container complaint_management_system-kafka-1 Created 1.4s
✓ Container complaint_management_system-kong-1 Created 3.7s
✓ Container complaint_management_system-fastapi-1 Creat... 5.6s
✓ Container complaint_management_system-streamlit-1 Cre... 5.5s
Attaching to fastapi-1, kafka-1, kong-1, kong-db-1, postgres-1, streamlit-1
kafka-1 | ===> User
kafka-1 | uid=1000(appuser) gid=1000(appuser) groups=1000(appuser)
kafka-1 | ===> Configuring ...
kafka-1 | Running in KRaft mode...
postgres-1 | The files belonging to this database system will be owned by user "postgres".
postgres-1 | This user must also own the server process.
kong-db-1 | The files belonging to this database system will be owned by user "postgres".
postgres-1 |
kong-db-1 | This user must also own the server process.
postgres-1 | The database cluster will be initialized with locale "en_US.utf8".
kong-db-1 |
postgres-1 | The default database encoding has accordingly been set to "UTF8".
kong-db-1 | The database cluster will be initialized with locale "en_US.utf8".
postgres-1 | The default text search configuration will be set to "english".
kong-db-1 | The default database encoding has accordingly been set to "UTF8".
postgres-1 |
kong-db-1 | The default text search configuration will be set to "english".
postgres-1 | Data page checksums are disabled.
kong-db-1 |
postgres-1 |
kong-db-1 | Data page checksums are disabled.
postgres-1 | fixing permissions on existing directory /var/lib/postgresql/data ... ok
kong-db-1 |


docker desktop
Search: bitnami/kafka
Ctrl+K
Sign in
Containers
Images
Volumes
Builds
Docker Scout
Extensions
Containers
Give feedback
Container CPU usage
0.25% / 400% (4 CPUs available)
Container memory usage
70.99MB / 3.64GB
Show charts
Search
Only show running containers
Name
Image
Status
Port(s)
CPU (%)
Last started
Actions
complaint_management
Running (6/6)
0.19%
2 seconds ago
kafka-1
c9d473230b1e
confluentinc/cp-kaf
Running
9092:9092
0%
2 seconds ago
kong-db-1
98ca406a6861
postgres:13
Running
5433:5432
0.1%
18 seconds ago
postgres-1
1eece5f682c9
postgres:13
Running
5432:5432
0.09%
18 seconds ago
kong-1
02ed3f498fba
kong:latest
Running
8004:8000
Show all ports (4)
0%
2 seconds ago
fastapi-1
6c955b623b51
complaint_manage
Running
8003:8000
0%
2 seconds ago
Showing 7 items
Engine running
RAM 2.07 GB CPU 57.92%
v4.31.1


docker desktop
Search: bitnami/kafka
Ctrl+K
Sign in
Containers
Images
Volumes
Builds
Docker Scout
Extensions
complaint_management_system
D:\Complaint_Management_System
Open
complaint_manag...
confluentinc/cp-kafl
Running
9092:9092
complaint_manag...
postgres:13
Running
5433:5432
complaint_manag...
postgres:13
Running
5432:5432
complaint_manag...
kong:latest
Running
8004:8000
Show all ports (4)
complaint_manag...
complaint_manager
Running
8003:8000
complaint_manag...
complaint_manager
2024-06-28 03:02:48 kong-db-1 |
2024-06-28 03:02:48 kong-db-1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
2024-06-28 03:02:48 kong-db-1 |
2024-06-28 02:58:45 postgres-1 | fixing permissions on existing directory /var/lib/postgresql/data ... ok
2024-06-28 02:58:45 postgres-1 | creating subdirectories ... ok
2024-06-28 02:58:45 postgres-1 | selecting dynamic shared memory implementation ... posix
2024-06-28 02:58:46 postgres-1 | selecting default max_connections ... 100
2024-06-28 02:58:46 postgres-1 | selecting default shared_buffers ... 128MB
2024-06-28 02:58:47 postgres-1 | selecting default time zone ... Etc/UTC
2024-06-28 02:58:47 postgres-1 | creating configuration files ... ok
2024-06-28 02:58:49 postgres-1 | running bootstrap script ... ok
2024-06-28 03:02:48 kong-db-1 | 2024-06-27 22:02:48.843 UTC [1] LOG: starting PostgreSQL 13.15 (Debian 13.15-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
2024-06-28 03:02:48 kong-db-1 | 2024-06-27 22:02:48.850 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
2024-06-28 03:02:48 kong-db-1 | 2024-06-27 22:02:48.856 UTC [1] LOG: listening on IPv6 address "::", port 5432
2024-06-28 03:02:48 kong-db-1 | 2024-06-27 22:02:48.879 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-06-28 03:02:48 kong-db-1 | 2024-06-27 22:02:48.921 UTC [27] LOG: database system was shut down at 2024-06-27 22:02:46 UTC
2024-06-28 03:02:49 kong-db-1 | 2024-06-27 22:02:49.010 UTC [1] LOG: database system is ready to accept connections
2024-06-28 02:58:56 postgres-1 | performing post-bootstrap initialization ... ok
2024-06-28 02:58:59 postgres-1 | syncing data to disk ... ok
2024-06-28 02:58:59 postgres-1 |
Engine running
RAM 2.12 GB CPU 82.85%
v4.31.1
1


streamlit_app
localhost:8501
Import favorites
Gmail
YouTube
Python W3 school
ChatGPT
re — Regular expre...
DLAI - Learning Pla...
Google Classroom
Deploy
Complaint Management System
Username
Password
Sign Up
Log In