

LAPORAN PRAKTIKUM

INTERNET OF THINGS

Tugas Simulator.py



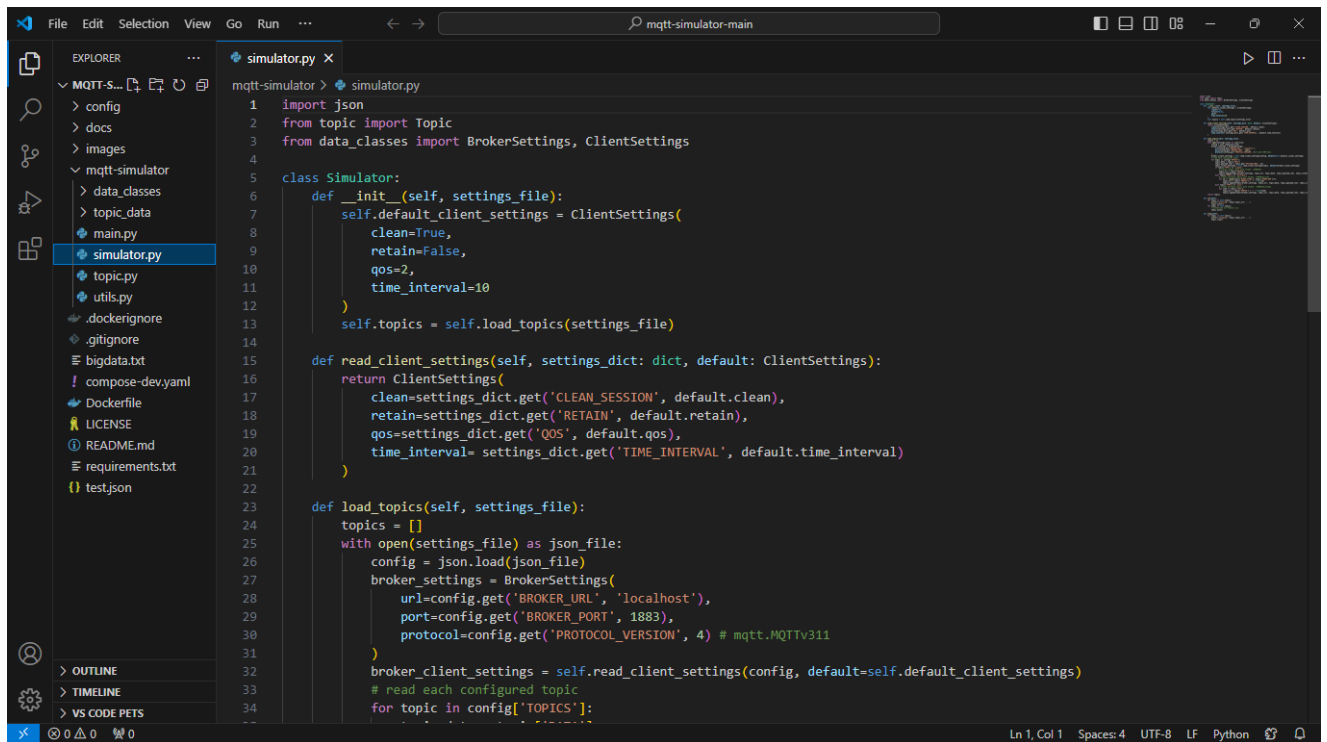
Nama : Jessica Natasya Sibarani

NIM : 11323048

Prodi : D3 – Teknologi Informasi

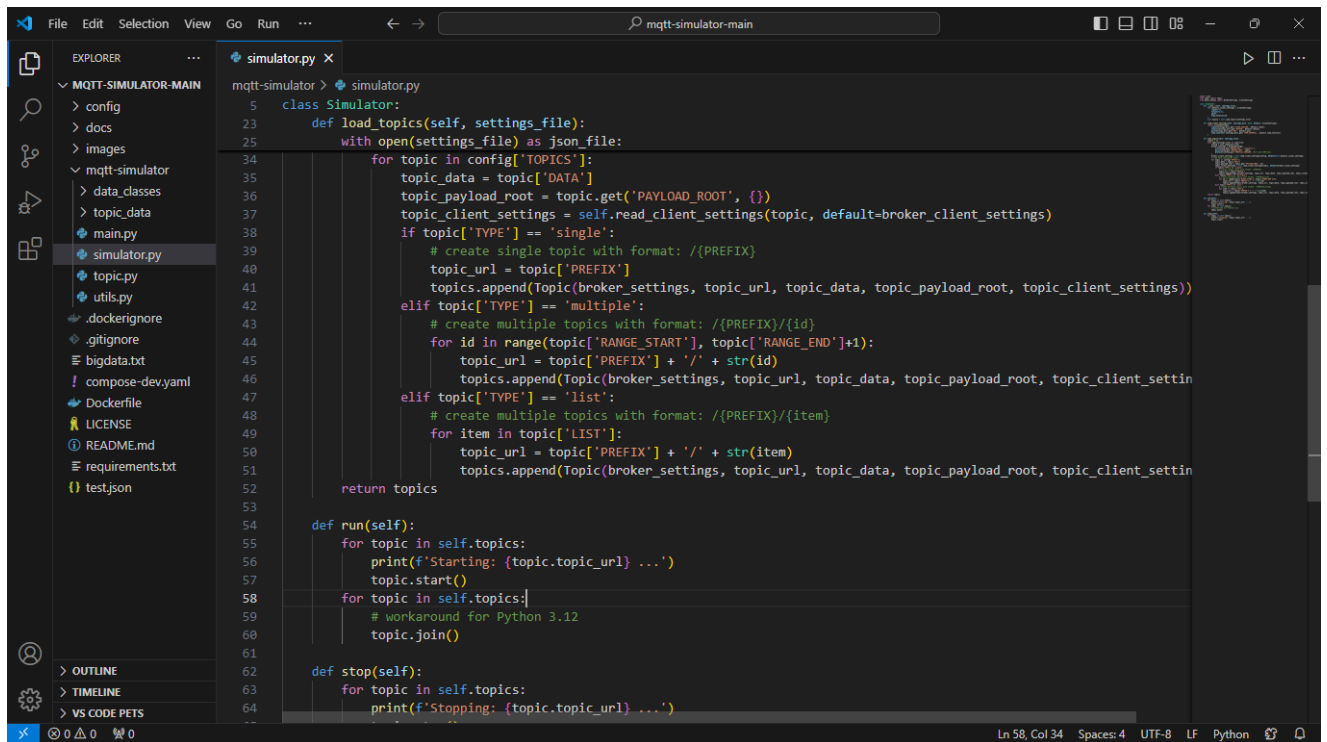
INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI
TAHUN AJARAN 2024/2025

Code program:



This screenshot shows the first 34 lines of the `simulator.py` file in a VS Code editor. The Explorer sidebar on the left shows the project structure for `MQTT-SIMULATOR-MAIN`, with `simulator.py` selected. The code defines a `Simulator` class with an `__init__` method that sets default client settings and loads topics from a settings file. It also includes methods for reading client settings and loading topics.

```
1 import json
2 from topic import Topic
3 from data_classes import BrokerSettings, ClientSettings
4
5 class Simulator:
6     def __init__(self, settings_file):
7         self.default_client_settings = ClientSettings(
8             clean=True,
9             retain=False,
10             qos=2,
11             time_interval=10
12         )
13         self.topics = self.load_topics(settings_file)
14
15     def read_client_settings(self, settings_dict: dict, default: ClientSettings):
16         return ClientSettings(
17             clean=settings_dict.get('CLEAN_SESSION', default.clean),
18             retain=settings_dict.get('RETAIN', default.retain),
19             qos=settings_dict.get('QOS', default.qos),
20             time_interval=settings_dict.get('TIME_INTERVAL', default.time_interval)
21         )
22
23     def load_topics(self, settings_file):
24         topics = []
25         with open(settings_file) as json_file:
26             config = json.load(json_file)
27             broker_settings = BrokerSettings(
28                 url=config.get('BROKER_URL', 'localhost'),
29                 port=config.get('BROKER_PORT', 1883),
30                 protocol=config.get('PROTOCOL_VERSION', 4) # mqtt.MQTTv311
31             )
32             broker_client_settings = self.read_client_settings(config, default=self.default_client_settings)
33             # read each configured topic
34             for topic in config['TOPICS']:
```



This screenshot shows the continuation of the `simulator.py` file, from line 34 to line 64. The `load_topics` method continues with logic to process each topic configuration, including handling single, multiple, and list topics. The `run` and `stop` methods are also defined.

```
34         for topic in config['TOPICS']:
35             topic_data = topic['DATA']
36             topic_payload_root = topic.get('PAYLOAD_ROOT', {})
37             topic_client_settings = self.read_client_settings(topic, default=broker_client_settings)
38             if topic['TYPE'] == 'single':
39                 # create single topic with format: /(PREFIX)
40                 topic_url = topic['PREFIX']
41                 topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
42             elif topic['TYPE'] == 'multiple':
43                 # create multiple topics with format: /(PREFIX)/(id)
44                 for id in range(topic['RANGE_START'], topic['RANGE_END']+1):
45                     topic_url = topic['PREFIX'] + '/' + str(id)
46                     topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
47             elif topic['TYPE'] == 'list':
48                 # create multiple topics with format: /(PREFIX)/(item)
49                 for item in topic['LIST']:
50                     topic_url = topic['PREFIX'] + '/' + str(item)
51                     topics.append(Topic(broker_settings, topic_url, topic_data, topic_payload_root, topic_client_settings))
52         return topics
53
54     def run(self):
55         for topic in self.topics:
56             print(f'Starting: {topic.topic_url} ...')
57             topic.start()
58         for topic in self.topics:
59             # workaround for Python 3.12
60             topic.join()
61
62     def stop(self):
63         for topic in self.topics:
64             print(f'Stopping: {topic.topic_url} ...')
```

Penjelasan:

Kode ini adalah implementasi Python untuk sebuah simulator yang bekerja dengan pengaturan broker MQTT dan daftar topik yang didefinisikan dalam file JSON. Berikut poin-poin utamanya:

1. Impor Modul dan Kelas

- Menggunakan modul json untuk membaca file konfigurasi JSON.
- Kelas Topic, BrokerSettings, dan ClientSettings digunakan untuk menangani topik serta pengaturan broker dan klien.

2. Kelas Simulator

- `__init__`: Menginisialisasi simulator dengan pengaturan klien bawaan dan memuat topik dari file JSON.
- `read_client_settings`: Membaca pengaturan klien MQTT dari konfigurasi JSON, dengan nilai bawaan sebagai fallback jika atribut tertentu tidak tersedia.
- `load_topics`: Membaca file JSON, membuat pengaturan broker dan klien, lalu membangun daftar Topic berdasarkan tipe (single, multiple, atau list) dengan URL yang dihasilkan sesuai pengaturan.
- `run`: Memulai semua topik dalam daftar, mencetak informasi saat memulai, lalu memastikan eksekusi berjalan sampai selesai.
- `stop`: Menghentikan semua topik dan mencetak pesan saat masing-masing topik dihentikan.

3. Konfigurasi JSON

- File JSON memuat informasi seperti URL broker, port, versi protokol, serta daftar topik (TOPICS) dengan berbagai jenis konfigurasi (tipe topik, data, payload, dll.).

4. Format Topik

- `single`: URL dengan format /PREFIX.
- `multiple`: URL dalam format /PREFIX/{id} untuk rentang ID tertentu.

- list: URL dalam format /PREFIX/{item} untuk elemen yang ada dalam daftar tertentu.

5. Fungsionalitas Utama

- Simulator membaca pengaturan, membangun topik, dan menjalankan atau menghentikan proses untuk setiap topik sesuai perintah.

Kode ini berguna untuk mensimulasikan interaksi dengan topik MQTT berdasarkan pengaturan yang fleksibel, mendukung publikasi atau langganan dengan cara yang terstruktur dan otomatis.