01076105, 01075106

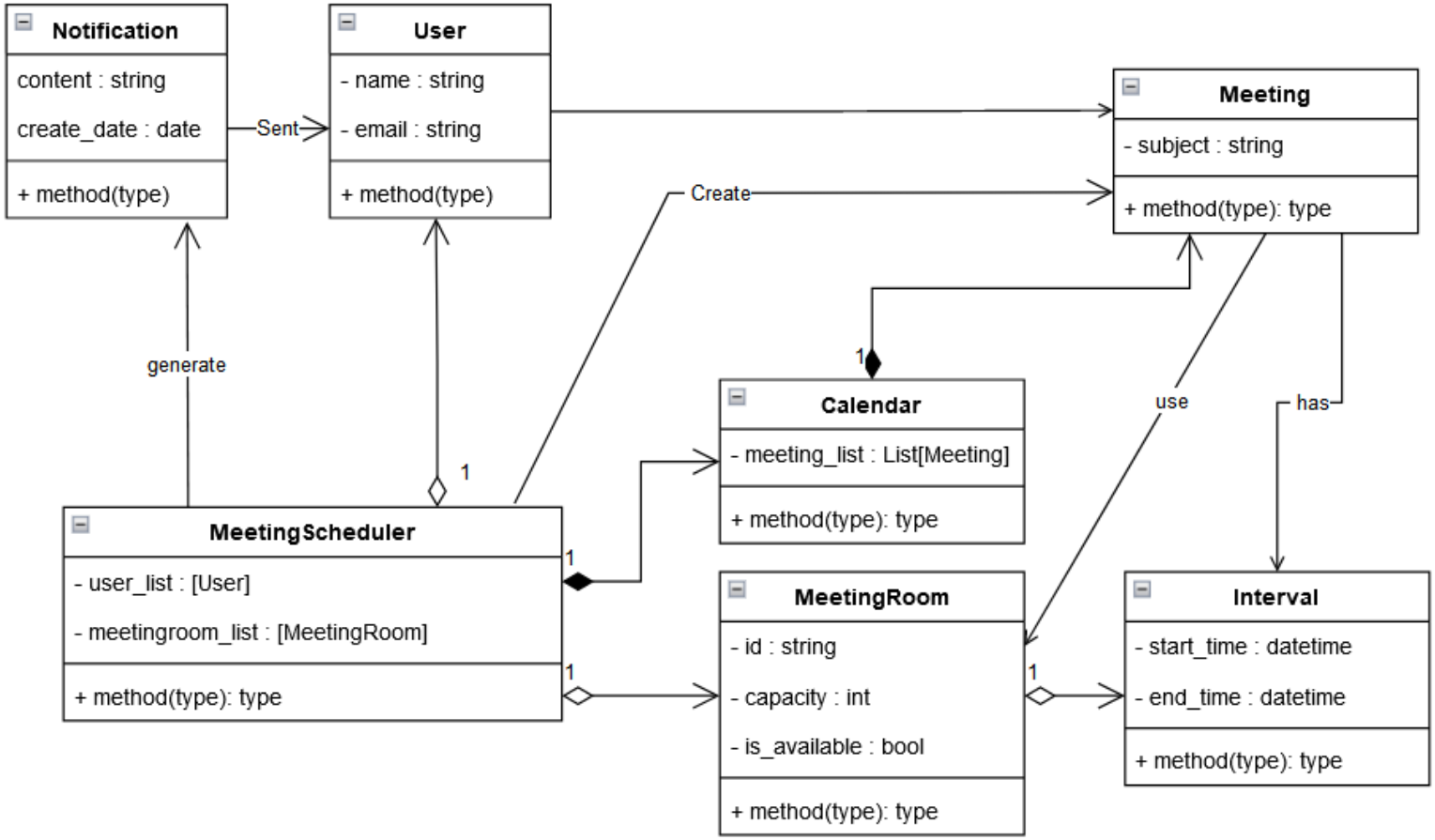
Object Oriented Programming

Object Oriented Programming Project

Special Lecture

Meeting Scheduler

- จาก Class Diagram จะสร้างคลาส และ method พื้นฐาน



Meeting Scheduler

- คลาส MeetingScheduler ทำหน้าที่เก็บ MeetingRoom และ User

```python
class MeetingScheduler:
    def __init__(self):
        self.__meeting_room_list = []
        self.__user_list = []

    def add_room(self, room):
        self.__meeting_room_list.append(room)

    def add_user(self, user):
        self.__user_list.append(user)

    def list_room(self):
        for i in self.__meeting_room_list:
            print(i)
```

Meeting Scheduler

- จากนั้นเพิ่มคลาส MeetingRoom โดยเป็นคลาสที่มีหลาย method

```python
class MeetingRoom:
    def __init__(self, id, capacity):
        self.__id = id
        self.__capacity = capacity
        self.__is_available = 'Y'
        self.__interval_list = []

    def is_available(self):
        return self.__is_available

    def add_interval(self, interval):
        self.__interval_list.append(interval)
```

Meeting Scheduler

- จากนั้นเพิ่มคลาส MeetingRoom โดยเป็นคลาสที่มีหลาย method
- Method room_available ใช้สำหรับตรวจสอบว่าห้องว่างสำหรับเวลานั้นหรือไม่

```
def check_no_overlap(start_time1, end_time1, start_time2, end_time2):
    if start_time1 > end_time2 or start_time2 > end_time1:
        return True
    else:
        return False

def room_available(self, datetime1, datetime2):
    for i in self.__interval_list:
        if not self.check_no_overlap(i.get_start_time, i.get_end_time, datetime1, datetime2):
            return False
    return True

def __str__(self):
    return(f"Room id : {self.__id} capacity {self.__capacity}")
```

Meeting Scheduler

- เนื่องจากใน MeetingRoom มีการใช้คลาส Interval

```python
class User:
    def __init__(self, name):
        self.__name = name

class Interval:
    def __init__(self, start_time, end_time):
        self.__start_time = start_time
        self.__end_time = end_time
```

Meeting Scheduler

- จากนั้นทดสอบสร้าง room และ user โปรแกรมทำงานได้

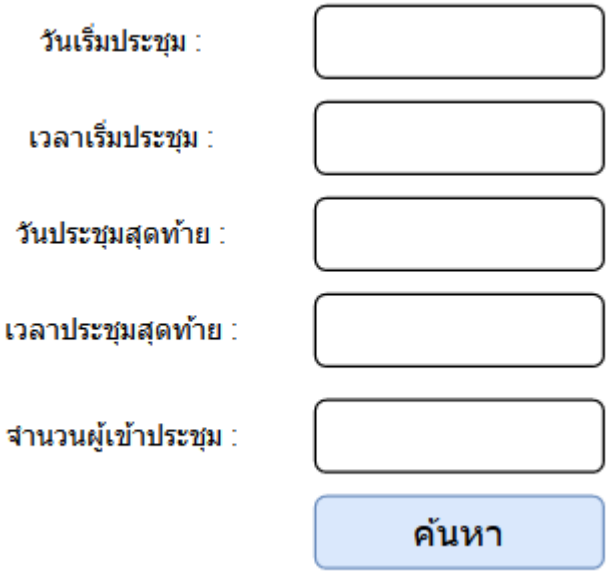
```
def main():
    meet = MeetingScheduler()
    for i in range(10):
        room_id = i + 1
        room_capacity = (i + 1) * 10
        meet.add_room(MeetingRoom(room_id, room_capacity))

    meet.list_room()
    john = User("john")

if __name__ == "__main__":
    main()
```

Meeting Scheduler

- คราวนี้จะกำหนด end point หรือ ส่วนที่เชื่อมต่อกับ frontend
  - End point สำหรับ ค้นหาห้องว่าง ใช้เมื่อจะจองห้อง โดยมี Interface ดังนี้
  - การส่งข้อมูลประกอบด้วย วัน เวลา เริ่ม วัน เวลา สุดท้าย และ จำนวน

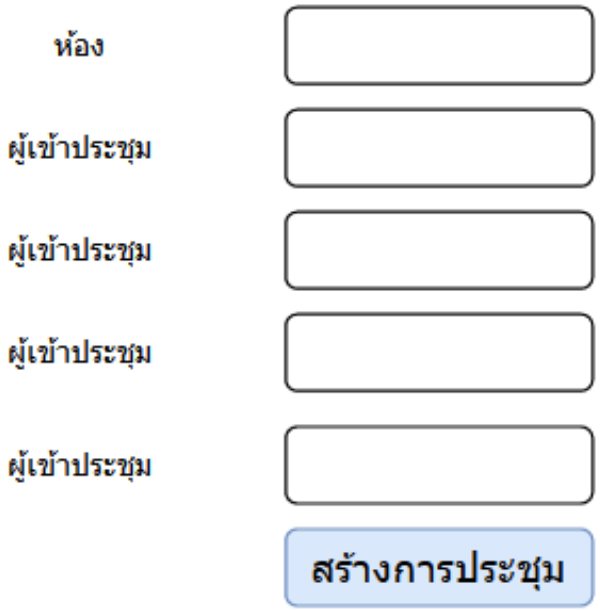


ค้นหาห้องประชุม
ต้องใช้งานได้
ต้องมีจำนวนเพียงพอ
ต้องว่างในช่วงวันเวลา

  - จะส่งกลับเป็น List ของห้องที่ว่างในเวลาดังกล่าว

Meeting Scheduler

- จะกำหนด end point หรือ ส่วนที่เชื่อมต่อกับ frontend
  - เมื่อได้ห้องแล้วจะสร้าง end point สำหรับการจองห้อง
  - การส่งข้อมูล ประกอบด้วย ห้อง และ List ของผู้เข้าร่วมประชุม
  - ผู้เข้าประชุม ขึ้นกับ frontend ว่าจะดำเนินการอย่างไร หน้านี้มีแค่ 4 เพื่อให้อธิบายง่าย



ดำเนินการสร้าง Meeting

เพิ่มผู้เข้าประชุมใน Meeting

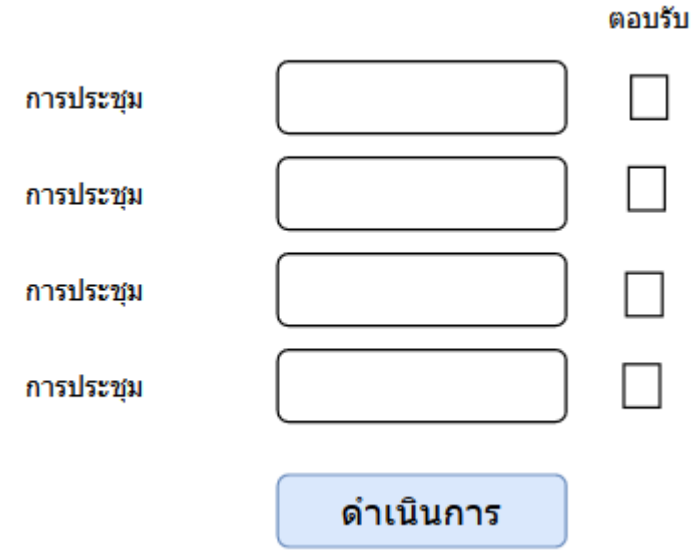
จัดเก็บ Meeting

สร้าง Notification

  - การส่งกลับ คือ status ว่าดำเนินการเรียบร้อยหรือไม่

Meeting Scheduler

- จะกำหนด end point หรือ ส่วนที่เชื่อมต่อกับ frontend
  - ผู้ใช้ตรวจสอบการประชุมของตนเอง และ ตอบรับ/ปฏิเสธเข้าประชุม
  - ในหน้านี้จะมี 2 endpoint คือ การค้นหาการประชุม และ การตอบรับ/ปฏิเสธ
  - Endpoint 1 จะส่ง user, endpoint2 จะส่ง user และ การประชุมที่ตอบรับ



นำ user ไปค้นหาการประชุม และส่งกลับ

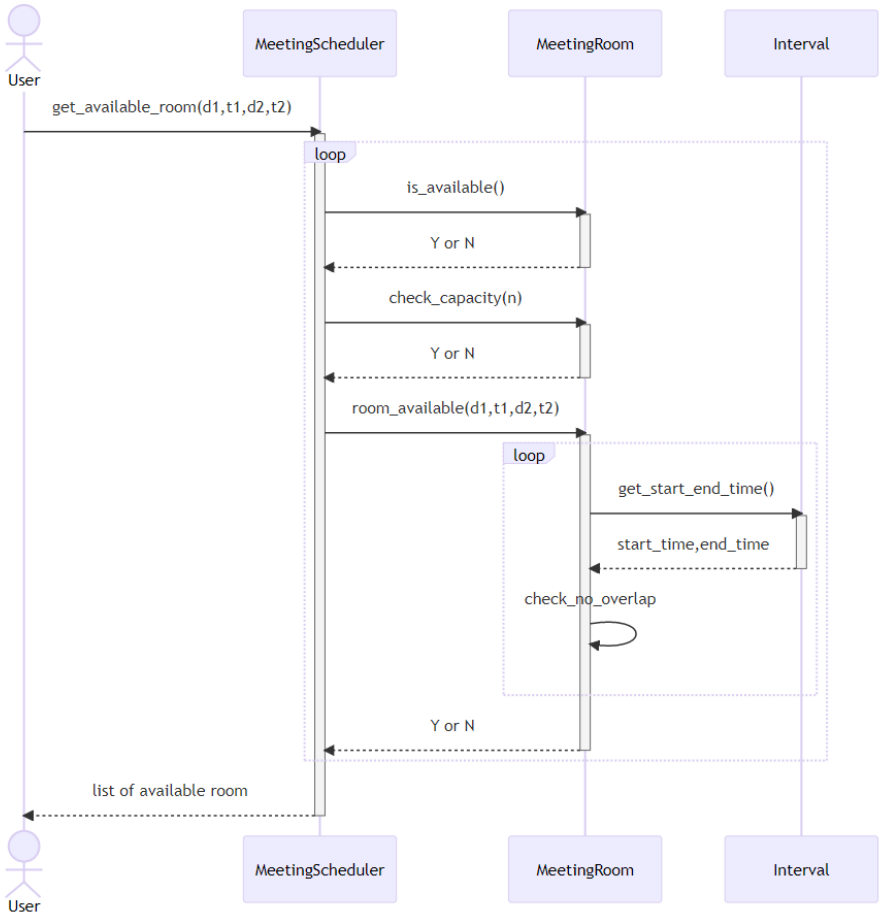
นำ user และ List การประชุม ไป add การตอบรับ

Meeting Scheduler

- อาจมี end point อื่นๆ
  - การสร้างห้อง
  - การสร้าง User

Meeting Scheduler

- คราวนี้จะสร้าง sequence diagram สำหรับ endpoint1



Meeting Scheduler

- จากนั้นนำ sequence diagram มาเขียนโปรแกรม

```
def find_available_room(self, start_date, start_time, end_date, end_time, capacity):
    date1 = datetime.strptime(start_date + " " + start_time, '%d-%m-%Y %H:%M')
    date2 = datetime.strptime(end_date + " " + end_time, '%d-%m-%Y %H:%M')
    available_room = []
    for i in self.__meeting_room_list:
        if not i.is_available:
            continue
        if i.capacity < capacity:
            continue
        if not i.room_available(date1, date2):
            continue
        available_room.append(i)
    return available_room
```

Meeting Scheduler

- จากนั้นทดสอบโปรแกรม โดยรันบรรทัดนี้

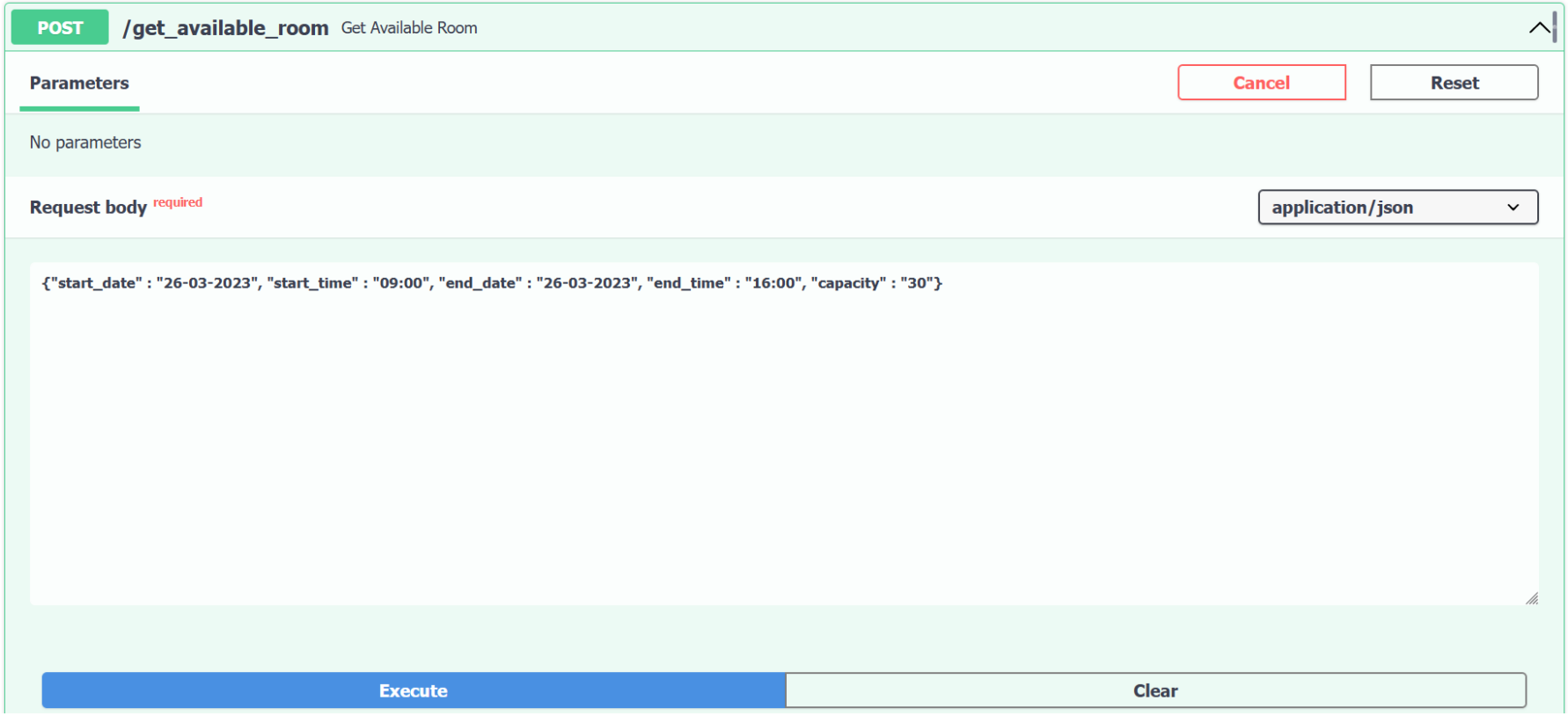
```
a_room = meet.find_available_room("26-03-2023", "09:00", "26-03-2023", "16:00", 30)
```

- เมื่อโปรแกรมใช้งานได้ ก็ทำเป็น Rest API เพื่อให้เรียกใช้ผ่าน API ได้
- ให้สังเกตว่าใน method ข้างต้นมีการส่งพารามิเตอร์เป็น string และ int ทั้งหมด ทั้งนี้เนื่องจากในการเรียกใช้ API จะสามารถส่งข้อมูลได้จำกัด
- ต่อไปจะสร้าง API ให้มาเรียกที่ method find_available_room

Meeting Scheduler

```python
# GET -- > Read Todo
@app.post("/get_available_room")
async def get_available_room(data:dict)->dict:
    st_d = data["start_date"]
    st_t = data["start_time"]
    end_d = data["end_date"]
    end_t = data["end_time"]
    capacity = int(data["capacity"])
    a_room = meet.find_available_room(st_d, st_t, end_d, end_t, capacity)
    for i in a_room:
        print(i)
    dt = {}
    for i in a_room:
        dt[i.id] = i.capacity
    print(dt)
    return {"Data": dt}
```

Meeting Scheduler



Meeting Scheduler

Request URL

```
http://127.0.0.1:8000/get_available_room
```

Server response

Code	Details
200	**Response body**

```
{
  "Data": {
    "3": 30,
    "4": 40,
    "5": 50,
    "6": 60,
    "7": 70,
    "8": 80,
    "9": 90,
    "10": 100
  }
}
```

Download

Response headers

```
content-length: 68
content-type: application/json
date: Sun,26 Mar 2023 08:34:18 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

For your attention