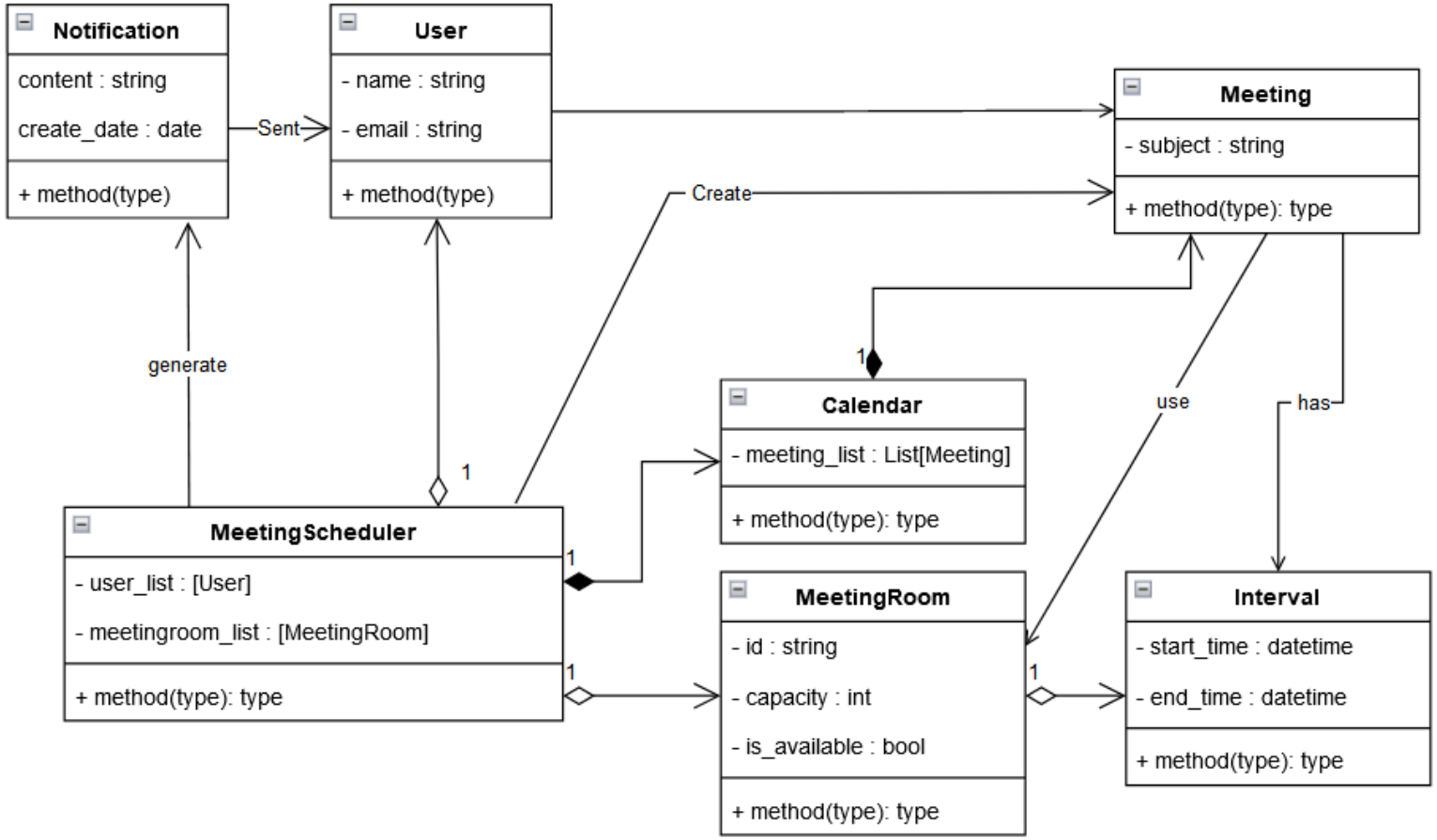
OOP App Development

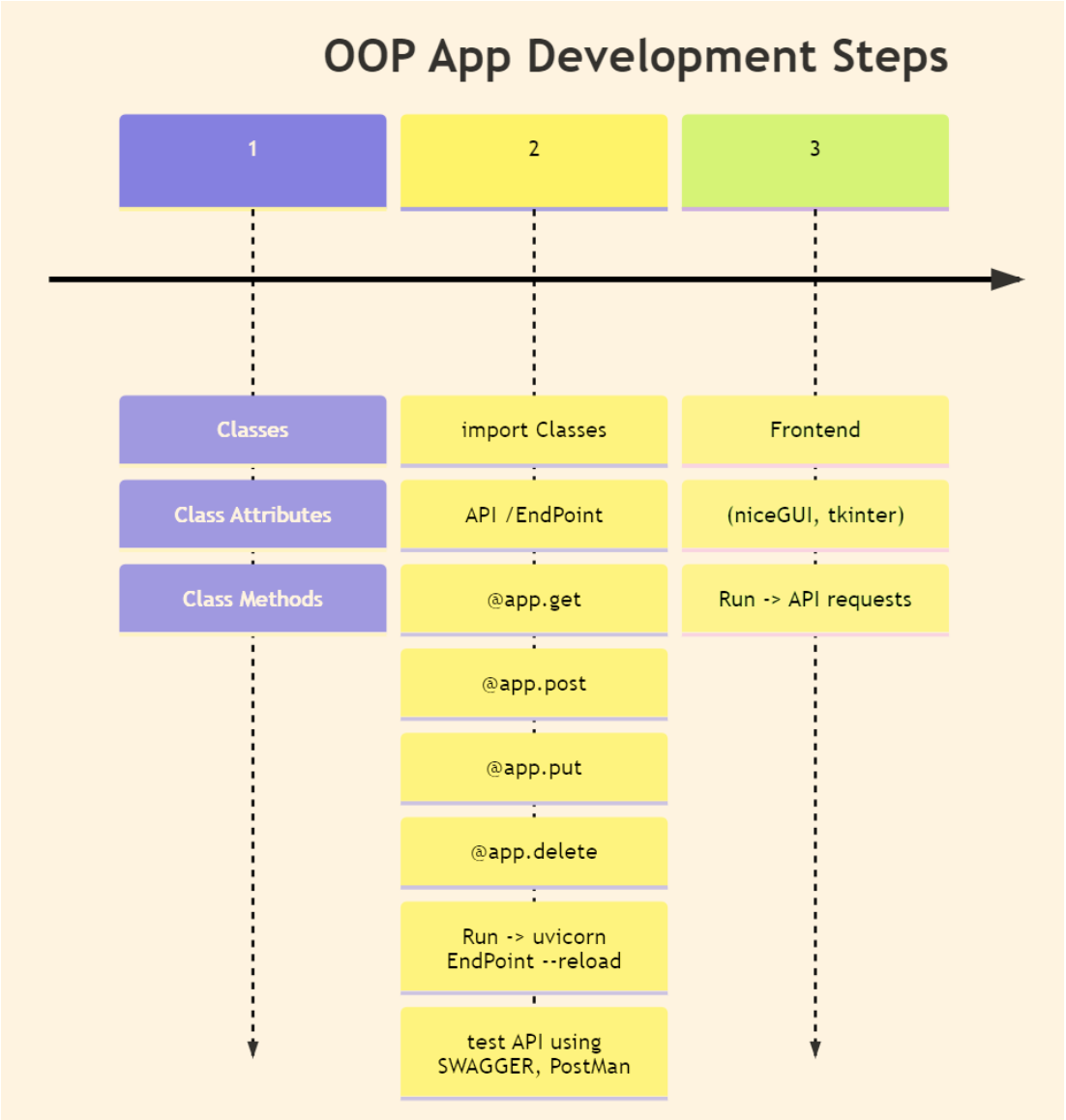
Code and Process Examples

Meeting Scheduler Class Diagram

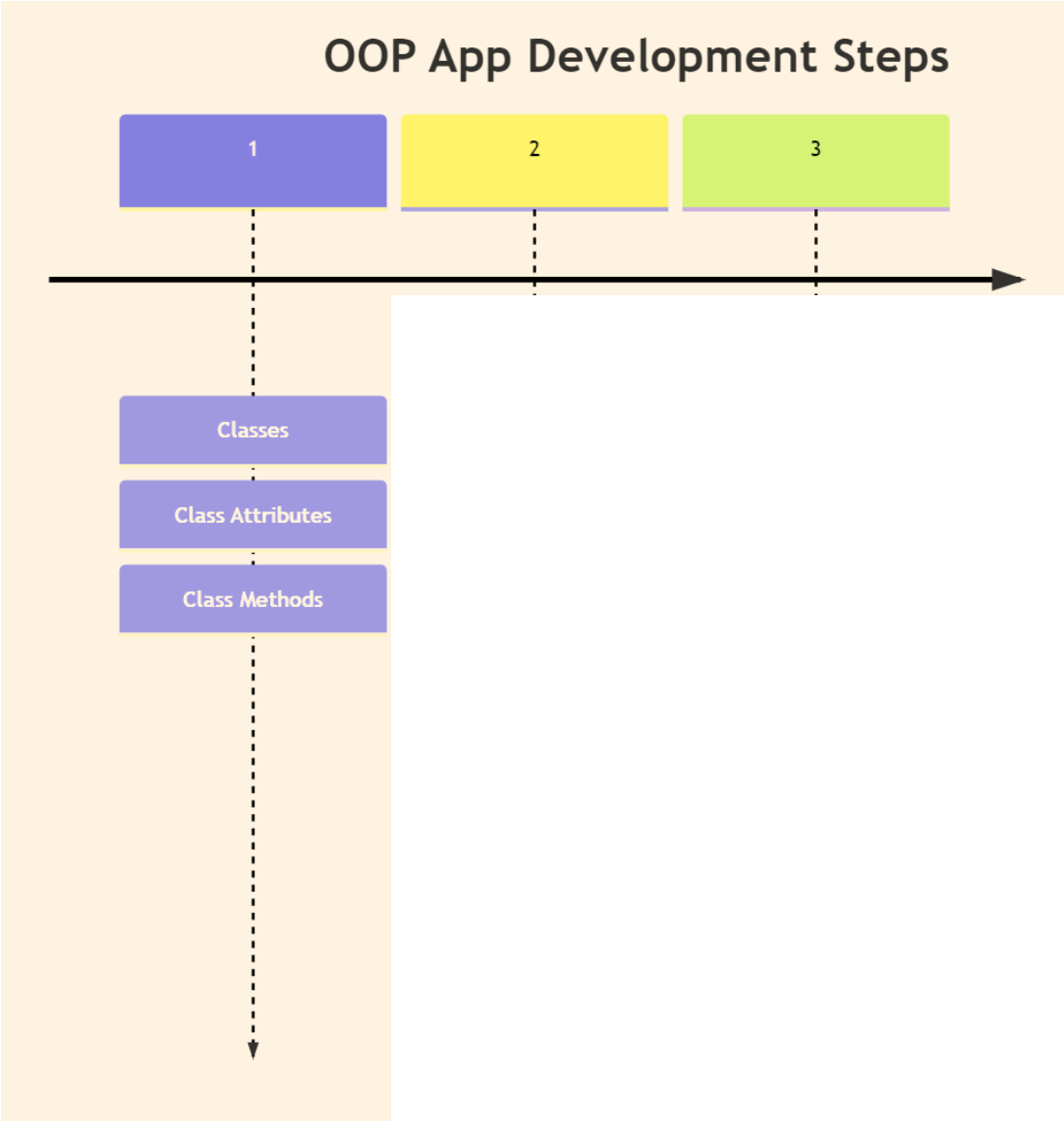
- จาก Class Diagram จะสร้างคลาส และ method พื้นฐาน

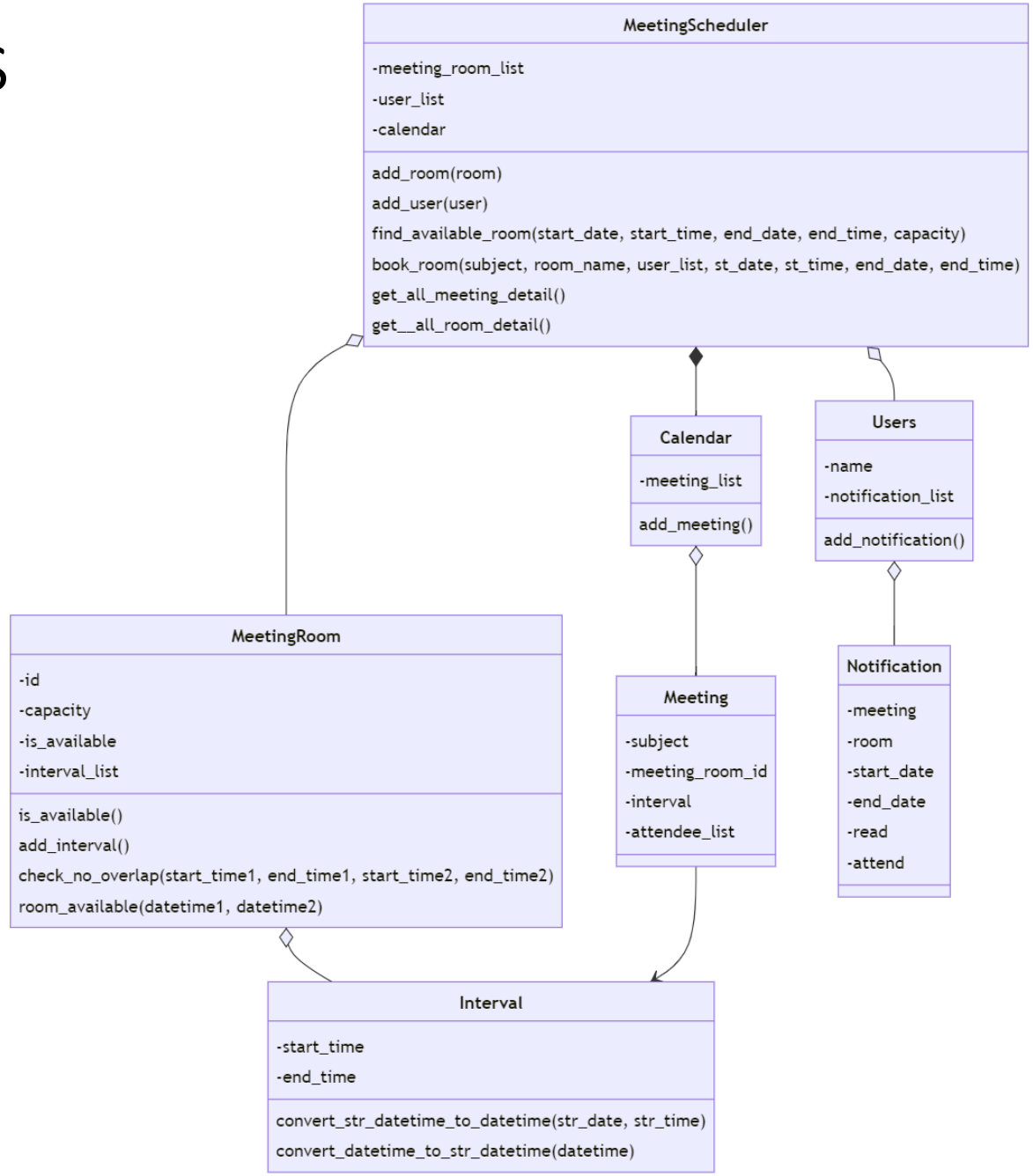


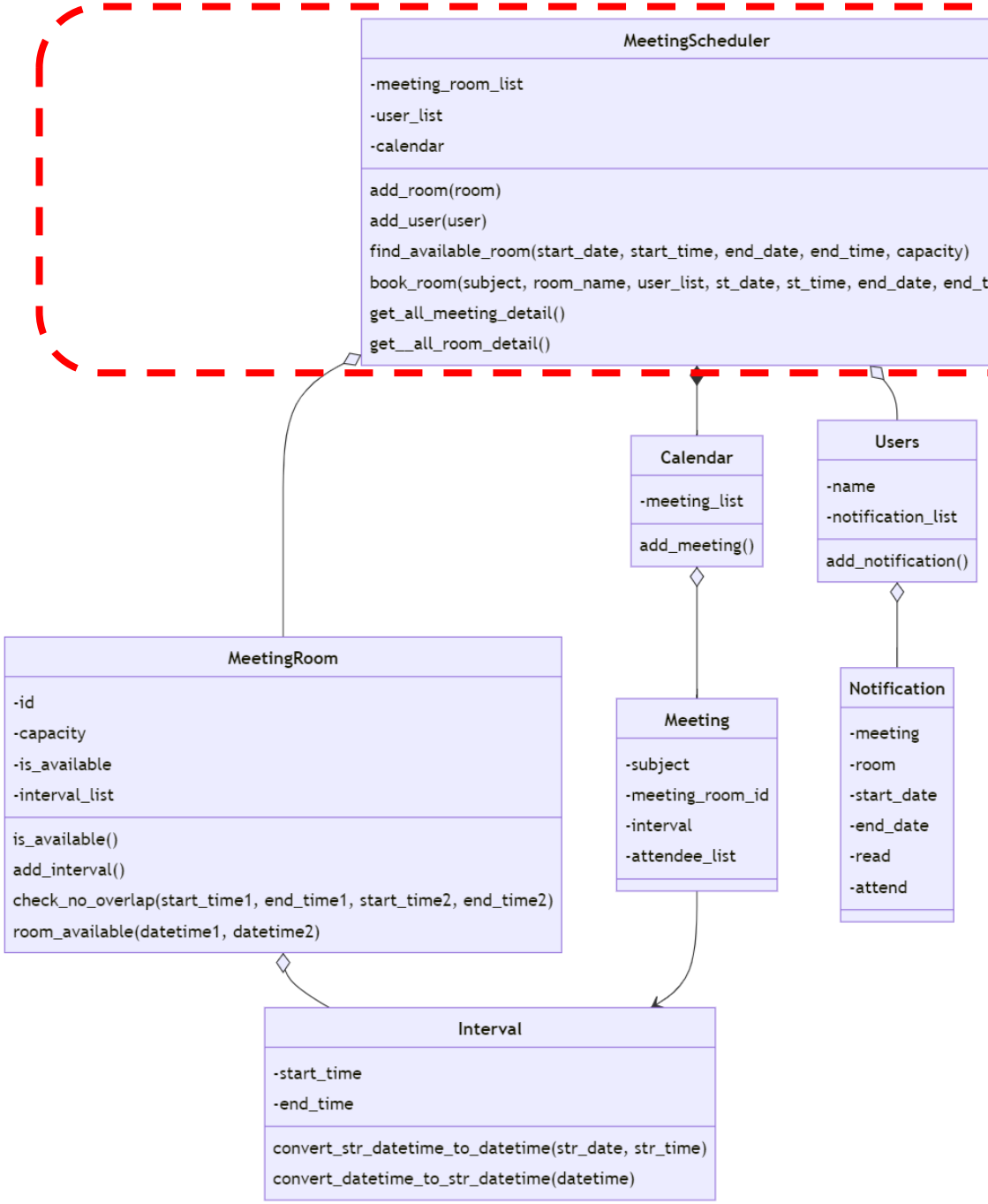
OOP App Development Steps



OOP App Development Steps







อาจเรียก class MeetingRoomScheduler
เป็น class System รวบรวมข้อมูลและให้บริการ (Service Method)

ทำหน้าที่รวบรวม List , Service Methods
- meeting_room_list
- user_list

class Users
เป็น class ดูแลข้อมูล name, notification_list

class Notification
เป็น class ดูแลข้อมูล แจ้งเตือนเกี่ยวกับการประชุม

class Meeting
เป็น class ดูแลข้อมูล เกี่ยวกับ Agenda การประชุม

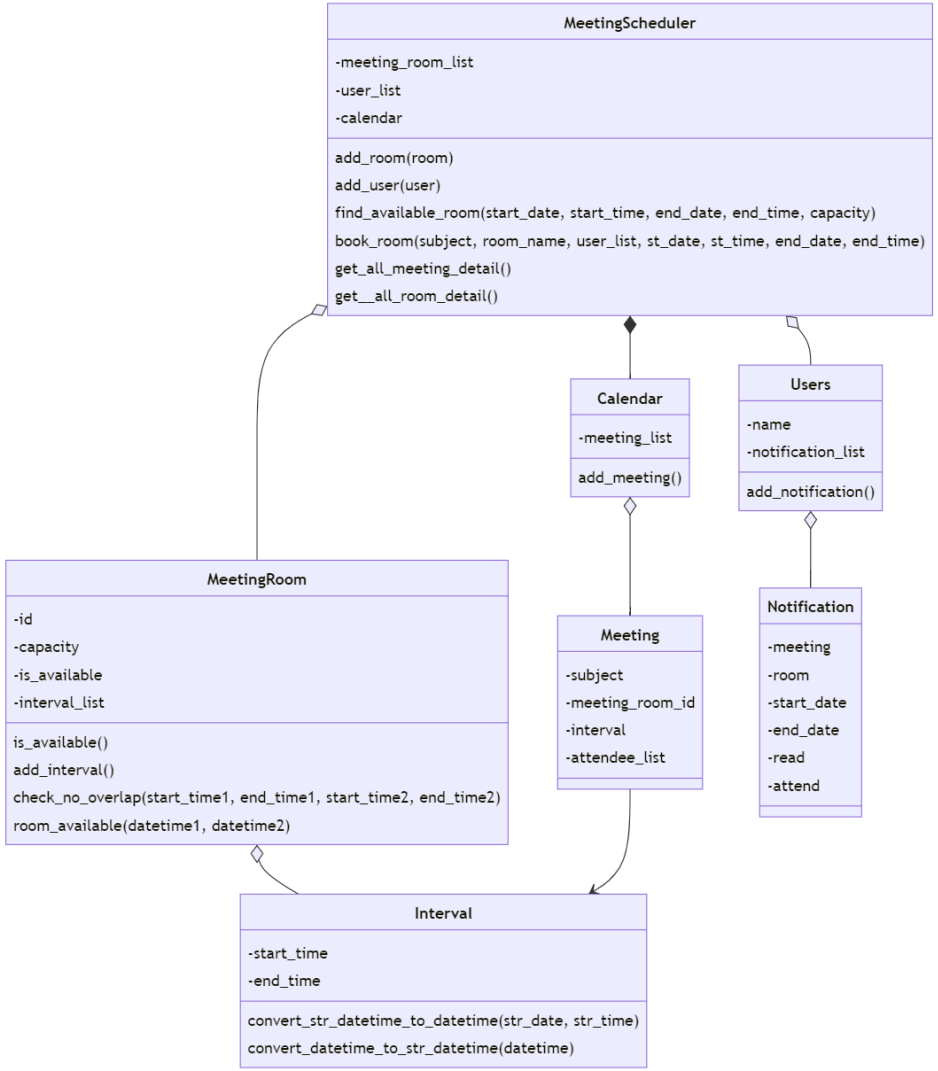
class MeetingRoom
เป็น class ดูแลข้อมูลห้อง จัดการเวลาจอง ตรวจสอบเวลาว่าง

class Interval
เป็น class ดูแลข้อมูลช่วงเวลาจอง

MeetingScheduler

ทำหน้าที่รวบรวม List , Service Methods

meeting_room_list, user_list, Calendar

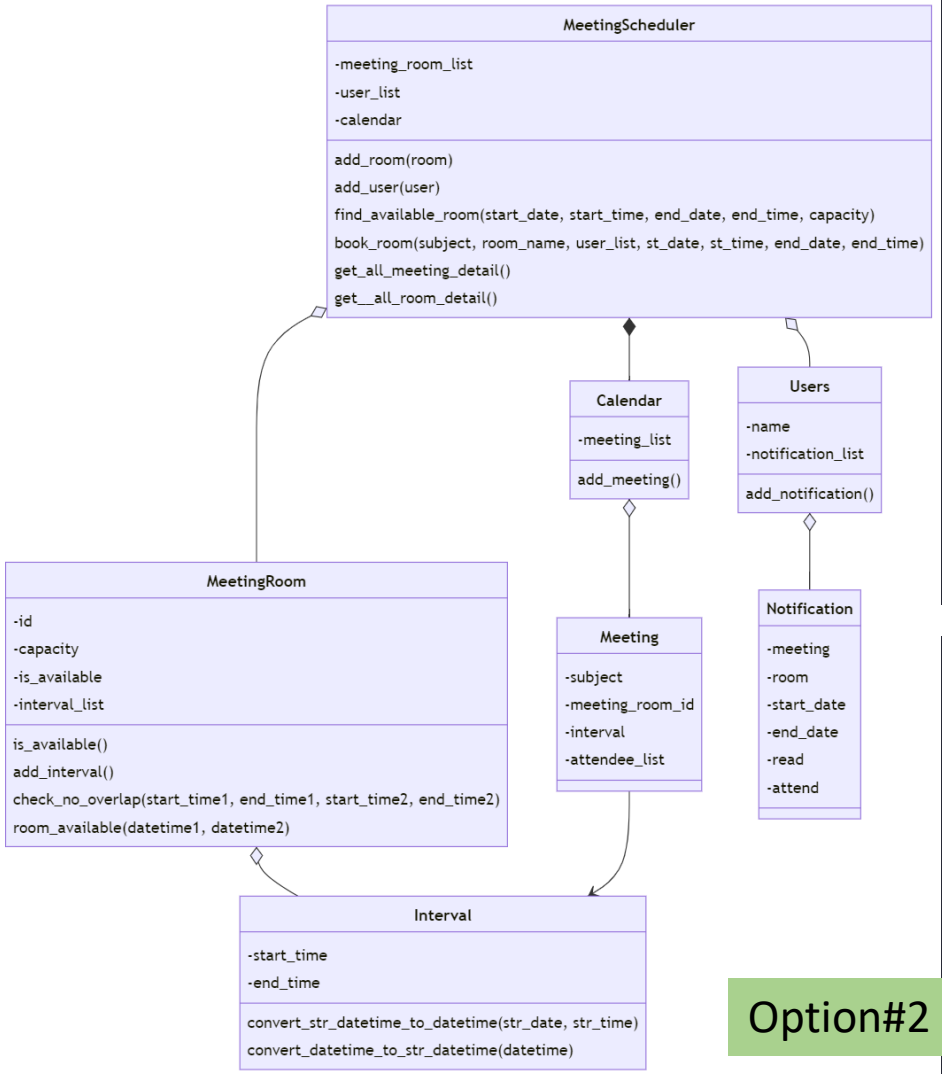


Option#1

```python
class MeetingScheduler:
    def __init__(self): ...

    def add_room(self, room): ...

    def add_user(self, user): ...

    def find_available_room(self, start_date, start_time, end_date, end_time, capacity): ...

    def book_room(self, subject, room_name, user_list, st_date, st_time, end_date, end_time): ...

    def show_meeting(self): ...

    def list_room(self): ...
```

Option#2

```python
class MeetingScheduler:
    def __init__(self): ...

    def add_room(self, room): ...

    def add_user(self, user): ...

    def add_calendar(self, meeting_calendar): ...

    def find_available_room(self, start_date, start_time, end_date, end_time, capacity): ...

    def book_room(self, meeting): ...

    def show_meeting(self): ...

    def list_room(self): ...
```




```python
def book_room(self, subject, room_name, user_list, st_date, st_time, end_date, end_time):
    o_user_list = []
    for user in self.__user_list:
        if user.name in user_list:
            o_user_list.append(user)
    for room in self.__meeting_room_list:
        if room.id == room_name:
            meeting_room = room
            break
    interval = Interval(st_date, st_time, end_date, end_time)
    meeting_room.add_interval(interval)
    meeting = Meeting(subject, meeting_room, interval, o_user_list)
    self.__calendar.add_meeting(meeting)
    noti = Notification(meeting, meeting_room, interval)
    for user in o_user_list:
        user.add_notification(noti)
    return "success"
```

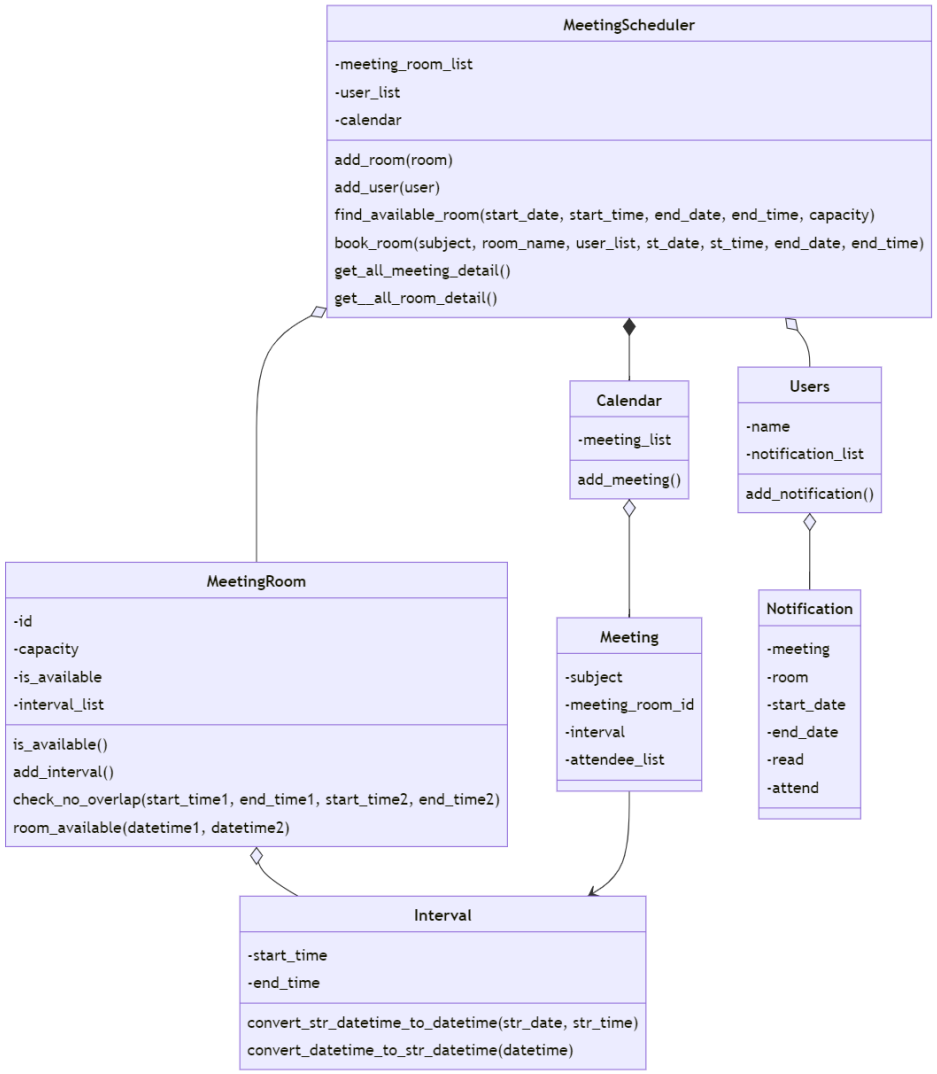
Option#1

Option#2

```python
def book_room(self, meeting):
    booking_room = None
    for room in self.__meeting_room_list:
        if room.id == meeting.meeting_room_id:
            booking_room = room
            break
    if booking_room is None:
        return "error: meeting room not found"

    booking_room.add_interval(meeting.interval)
    self.__calendar.add_meeting(meeting)
    noti = Notification(meeting)

    for attendee in meeting.attendee_list:
        attendee.add_notification(noti)
    return "success"
```

```python
class Meeting:
    def __init__(self, subject, meeting_room_name, interval, user_list):
        self.__subject = subject
        self.__meeting_room = meeting_room_name
        self.__interval = interval
        self.__user_list = user_list
```

```python
class Meeting:
    def __init__(self, subject, meeting_room_name, interval, attendee_list):
        self.__subject = subject
        self.__meeting_room_id = meeting_room_name
        self.__interval = interval
        self.__attendee_list = attendee_list
```




```python
def book_room(self, subject, room_name, user_list, st_date, st_time, end_date, end_time):
    o_user_list = []
    for user in self.__user_list:
        if user.name in user_list:
            o_user_list.append(user)
    for room in self.__meeting_room_list:
        if room.id == room_name:
            meeting_room = room
            break
    interval = Interval(st_date, st_time, end_date, end_time)
    meeting_room.add_interval(interval)
    meeting = Meeting(subject, meeting_room, interval, o_user_list)
    self.__calendar.add_meeting(meeting)
    noti = Notification(meeting, meeting_room, interval)
    for user in o_user_list:
        user.add_notification(noti)
    return "success"
```

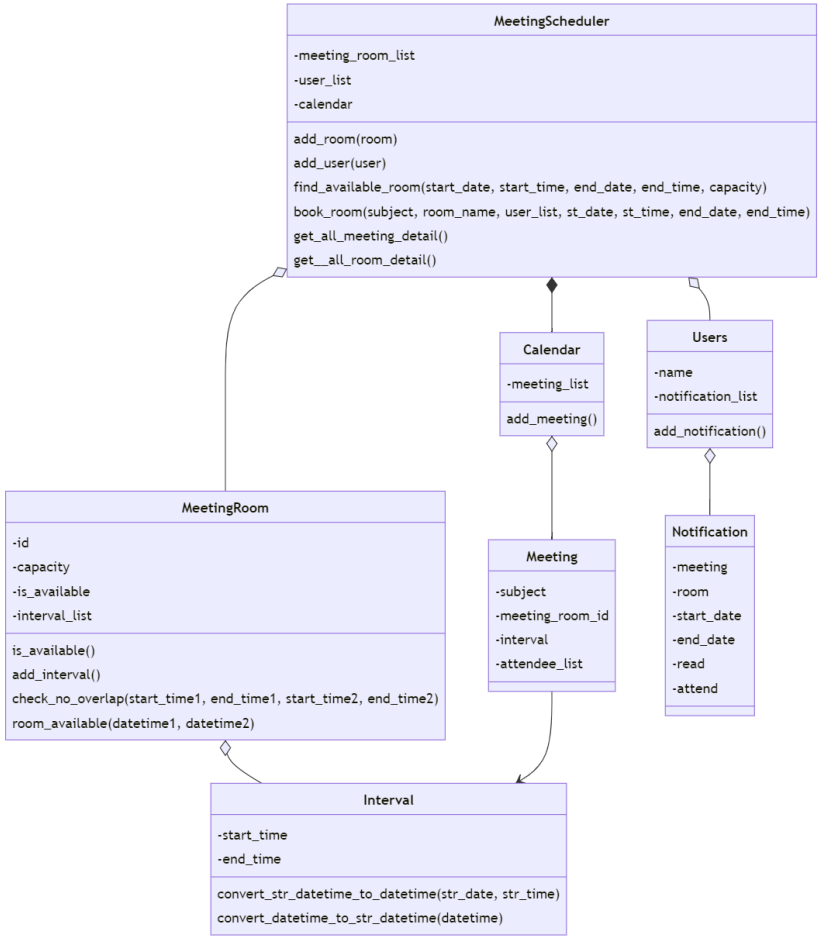
Option#1

```python
class Notification:
    def __init__(self, Meeting, room, interval):
        self.__meeting_subject = Meeting.subject
        self.__room = room.id
        self.__start_time = interval.start_time
        self.__end_time = interval.end_time
        self.__read = None
```

```python
def book_room(self, meeting):
    booking_room = None
    for room in self.__meeting_room_list:
        if room.id == meeting.meeting_room_id:
            booking_room = room
            break
    if booking_room is None:
        return "error: meeting room not found"

    booking_room.add_interval(meeting.interval)
    self.__calendar.add_meeting(meeting)
    noti = Notification(meeting)

    for attendee in meeting.attendee_list:
        attendee.add_notification(noti)
    return "success"
```

Option#2

```python
class Notification:
    def __init__(self, meeting):
        self.__meeting_subject = meeting.subject
        self.__room = meeting.meeting_room_id
        self.__start_time = meeting.interval.start_time
        self.__end_time = meeting.interval.end_time
        self.__read = None
```



Option#1

```python
class Notification:
    def __init__(self, Meeting, room, interval):
        self.__meeting_subject = Meeting.subject
        self.__room = room.id
        self.__start_time = interval.start_time
        self.__end_time = interval.end_time
        self.__read = None
```

Need getter()
using @property

to access
class private attribute

** สร้างเฉพาะจำเป็นต้องใช้

Option#2

```python
class Notification:
    def __init__(self, meeting):
        self.__meeting_subject = meeting.subject
        self.__room = meeting.meeting_room_id
        self.__start_time = meeting.interval.start_time
        self.__end_time = meeting.interval.end_time
        self.__read = None
```

```python
class MeetingRoom:
    def __init__(self, id, capacity):...

    def is_available(self):...

    @property
    def id(self):
        return self.__id

    @property
    def capacity(self):
        return self.__capacity

    def add_interval(self, interval):...

    def check_no_overlap(self,start_time1, end_time1, start_time2, end_time2):...

    def room_available(self, datetime1, datetime2):...

    def __str__(self):...
```

```python
class Meeting:
    def __init__(self, subject, meeting_room_name, interval, attendee_list):...

    @property
    def subject(self):
        return self.__subject

    @property
    def meeting_room_id(self):
        return self.__meeting_room_id

    @property
    def interval(self):
        return self.__interval

    @property
    def attendee_list(self):
        return self.__attendee_list

    def __str__(self):...
```

Option#1

book_room.py

```
def book_room(self, subject, room_name, user_list, st_date, st_time, end_date, end_time):
```

```
def test_book_room():
    meeting_manager.book_room("OOP Study", "5", ["john","alice"], "26-03-2023", "09:00", "26-03-2023", "16:00")
```

```
meeting_manager = MeetingScheduler()

# create 10 room instance for meeting_room_list
for i in range(10):
    room_id = i+1
    room_capacity = (i+1) * 10
    meeting_manager.add_room(MeetingRoom(str(room_id), room_capacity))

john = User("john")
meeting_manager.add_user(john)
meeting_manager.add_user(User("tom"))
meeting_manager.add_user(User("mary"))
meeting_manager.add_user(User("jenny"))
meeting_manager.add_user(User("bob"))
meeting_manager.add_user(User("alice"))

test_find_room_available()
test_book_room()
```

Class Declaration
book_room() in
class MeetingScheduler

Define test function

Object Instance Construction

Execute test function

Option#2

book_room.py

```python
def book_room(self, meeting):
```

```python
def test_book_room():
    booking_interval = Interval("26-03-2023", "09:00", "26-03-2023", "16:00")
    attendees = [john, tom]
    meeting_agenda = Meeting("OOP Study", "5", booking_interval, attendees)
    booking_status = meeting_manager.book_room(meeting_agenda)
```

```python
meeting_manager = MeetingScheduler()

# create 10 room instance for meeting_room_list
for i in range(10):
    room_id = i+1
    room_capacity = (i+1) * 10
    meeting_manager.add_room(MeetingRoom(str(room_id), room_capacity))

john = User("john")
tom = User("tom")
meeting_manager.add_user(john)
meeting_manager.add_user(tom)
meeting_manager.add_user(User("mary"))
meeting_manager.add_user(User("jenny"))
meeting_manager.add_user(User("bob"))
meeting_manager.add_user(User("alice"))

meeting_manager.add_calendar(Calendar())
```

```python
test_find_room_available()
test_book_room()
```

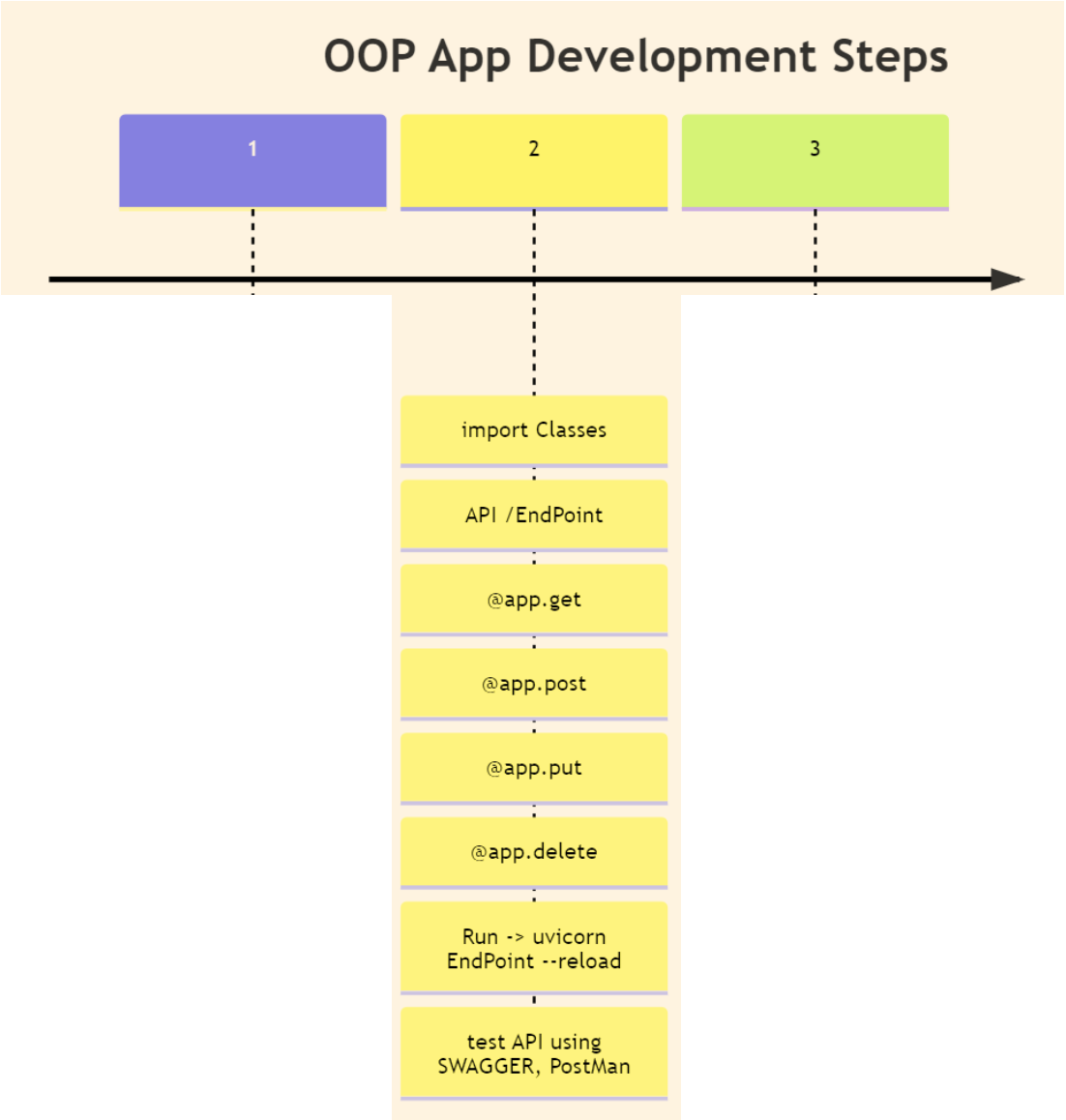
Class Declaration
book_room() in
class MeetingScheduler

Define test function

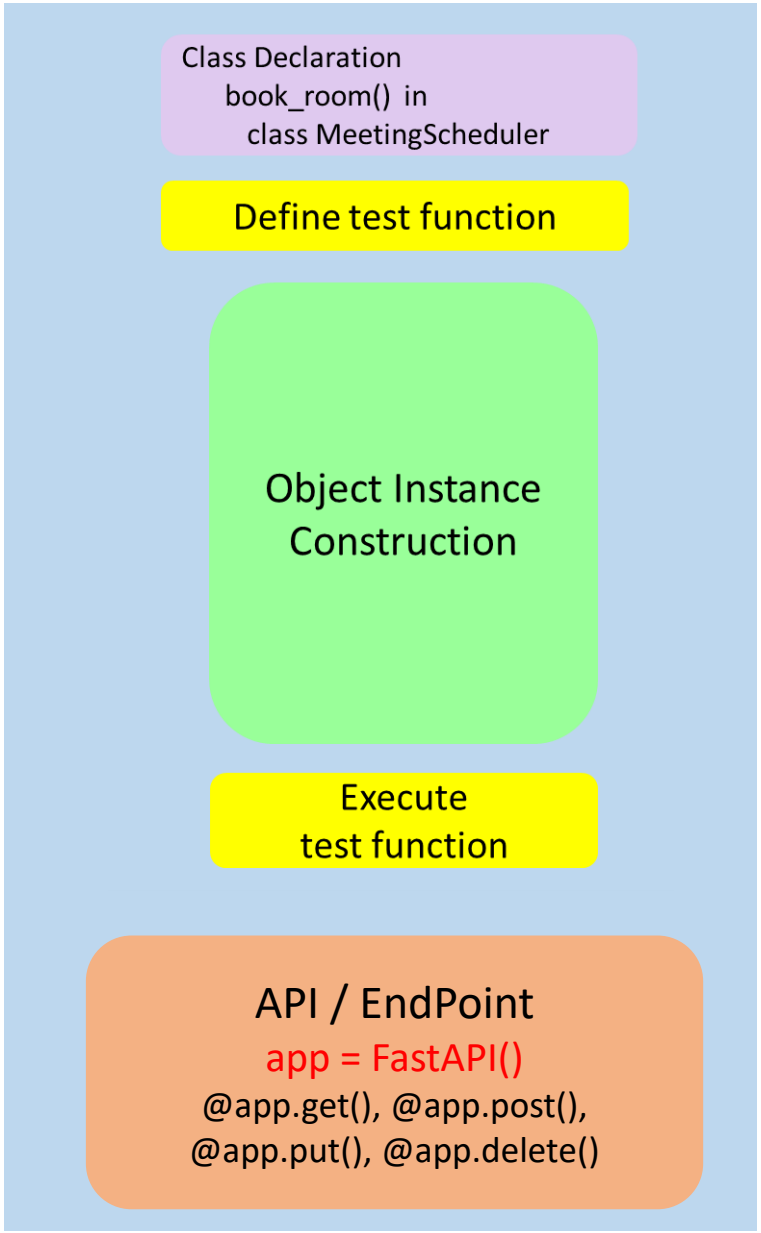
Object Instance Construction

Functional test

OOP App Development Steps



book_room.py



Test book_room.py : FastAPI

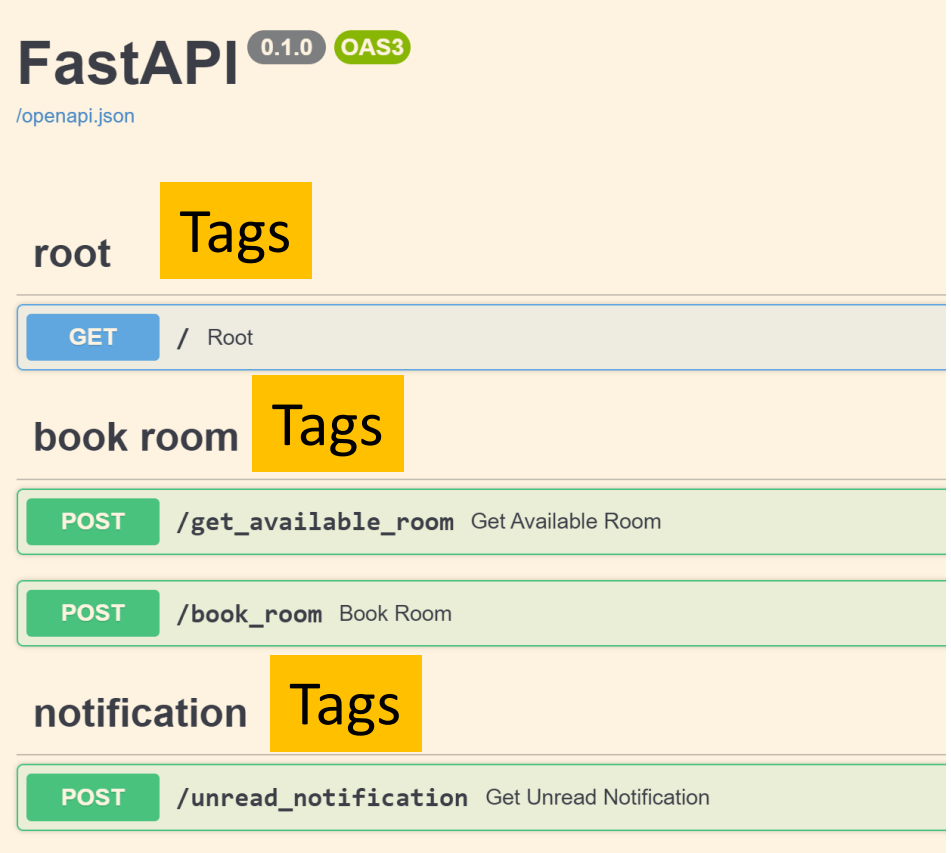
Tags -> จัดกลุ่ม API

```python
@app.post("/get_available_room", tags=["book room"])
async def get_available_room(data: dict) -> dict:...

@app.post("/book_room", tags=["book room"])
async def book_room(data: dict) -> dict:...

@app.post("/unread_notification", tags=["notification"])
async def get_unread_notification(data: dict) -> dict:...
```

Run on New Terminal
-> uvicorn book_room:app --reload

API Test

Swagger UI: http://127.0.0.1:8000/docs



Test book_room.py : FastAPI

Tags -> จัดกลุ่ม API

```
@app.post("/get_available_room", tags=["book room"])
async def get_available_room(data: dict) -> dict:···

@app.post("/book_room", tags=["book room"])
async def book_room(data: dict) -> dict:···

@app.post("/unread_notification", tags=["notification"])
async def get_unread_notification(data: dict) -> dict:···
```

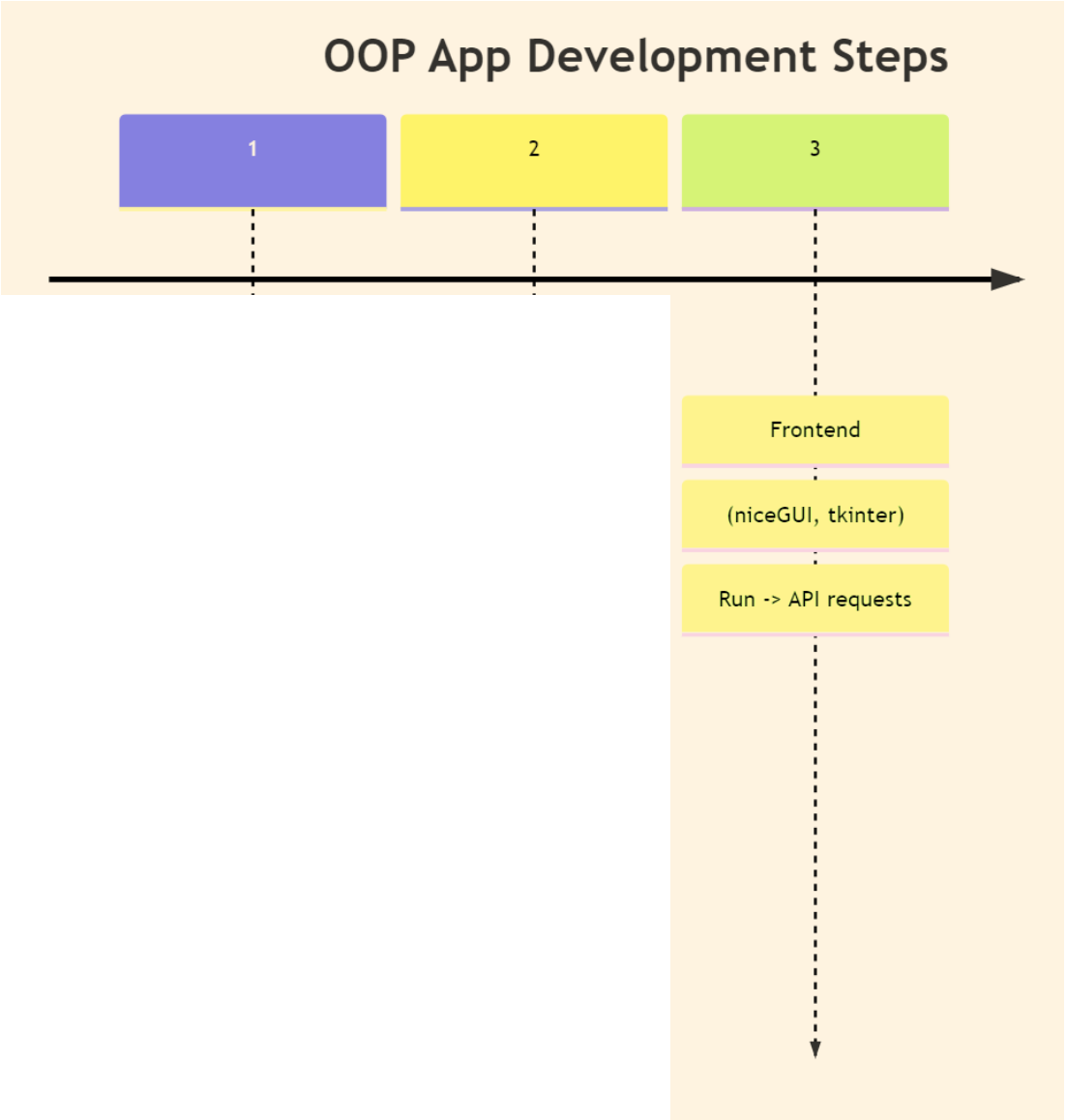
Run -> uvicorn book_room:app --reload

API Test

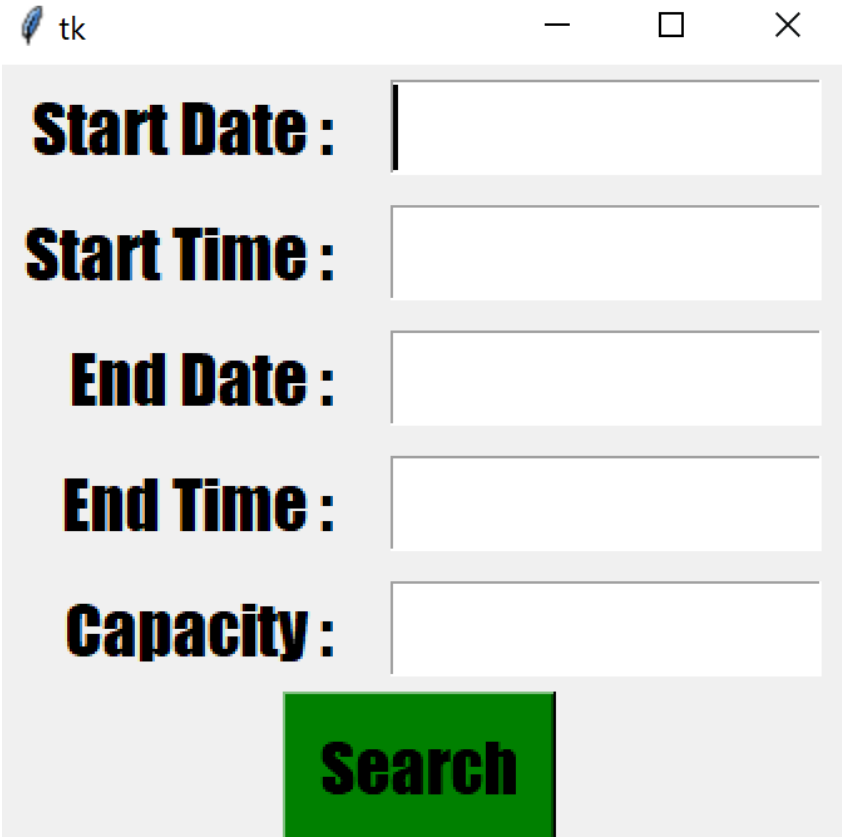
Request Library

```
r = requests.post("http://127.0.0.1:8000/get_available_room",data=json.dumps(room_filter))
print(r)
print(r.json())

r = requests.post("http://127.0.0.1:8000/book_room",data=json.dumps(meet_info))
print(r)
print(r.json())
```

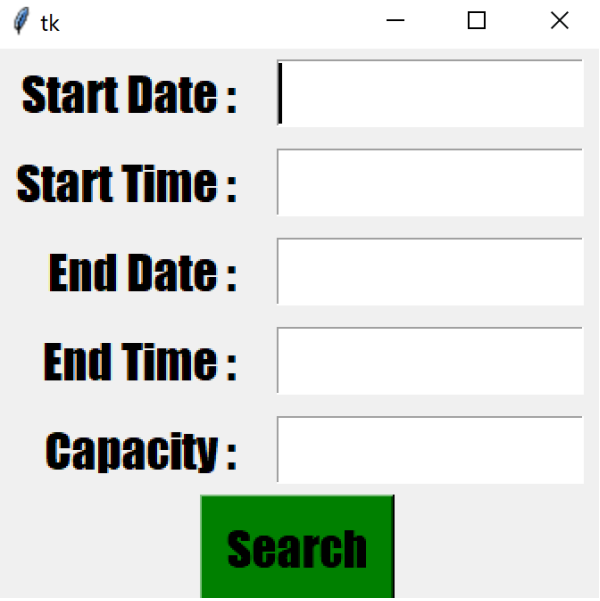
OOP App Development Steps



Write GUI: (Tkinter, NiceGUI, Frontend Framework)



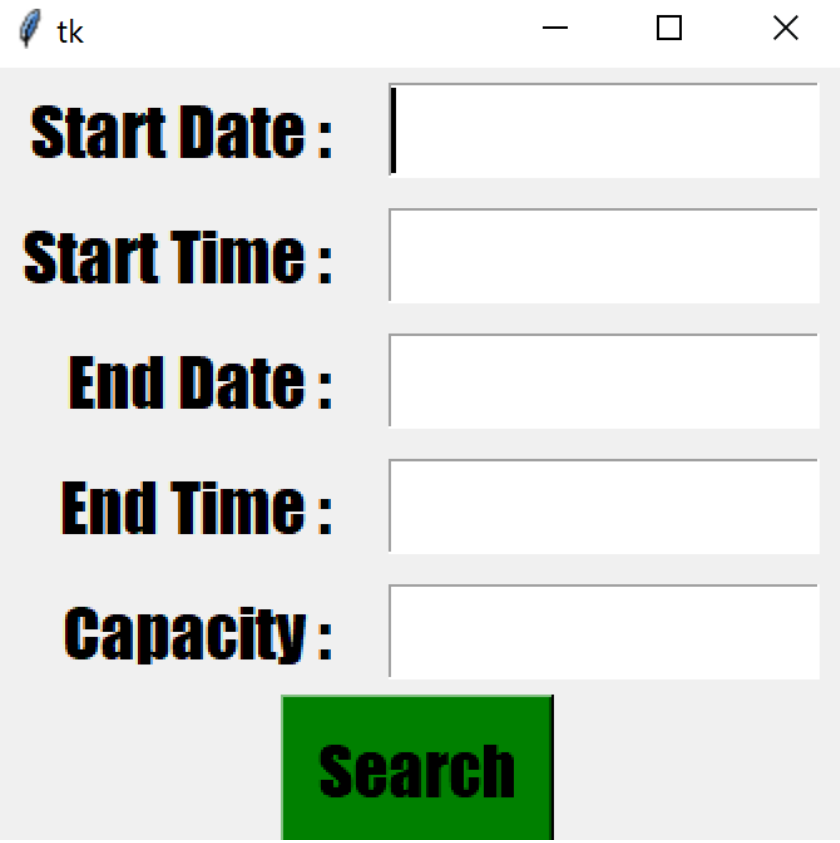
Tkinter GUI : tk_meet.py




```
root = Tk()
root.option_add("*Font", "impact 20")
st_date = StringVar()
st_time = StringVar()
end_date = StringVar()
end_time = StringVar()
capacity = StringVar()
select_opt = StringVar()
select_opt.set('3')
user1 = StringVar()
subject = StringVar()

Label(root, text="Start Date :").grid(row=0, column=0, padx=10, ipady=5, sticky='E')
Entry(root, textvariable=st_date, width=12, justify="left").grid(row=0, column=1, padx=10)
Label(root, text="Start Time :").grid(row=1, column=0,padx=10, ipady=5, sticky='E')
Entry(root, textvariable=st_time, width=12, justify="left").grid(row=1, column=1, padx=10)
Label(root, text="End Date :").grid(row=2, column=0,padx=10, ipady=5, sticky='E')
Entry(root, textvariable=end_date, width=12, justify="left").grid(row=2, column=1, padx=10)
Label(root, text="End Time :").grid(row=3, column=0,padx=10, ipady=5, sticky='E')
Entry(root, textvariable=end_time, width=12, justify="left").grid(row=3, column=1, padx=10)
Label(root, text="Capacity :").grid(row=4, column=0,padx=10, ipady=5, sticky='E')
Entry(root, textvariable=capacity, width=12, justify="left").grid(row=4, column=1, padx=10)
Button(root, text=" Search ", bg="green", command=on_click).grid(row=5, column=0, columnspan=2)
```

Tkinter GUI : tk_meet.py

UI ตัวนี้ ต้องการเชื่อม ENDPOINT 2 API




```python
from tkinter import *
import requests

API_ENDPOINT1 = "http://127.0.0.1:8000/get_available_room"
API_ENDPOINT2 = "http://127.0.0.1:8000/book_room"
```

Search Button -> on_click()

-> relate to API_ENDPOINT1

```python
def on_click():
    payload = {
        "start_date": st_date.get(),
        "start_time": st_time.get(),
        "end_date": end_date.get(),
        "end_time": end_time.get(),
        "capacity": capacity.get()
    }
    response = requests.post(API_ENDPOINT1, json=payload)
    if response.ok:
```

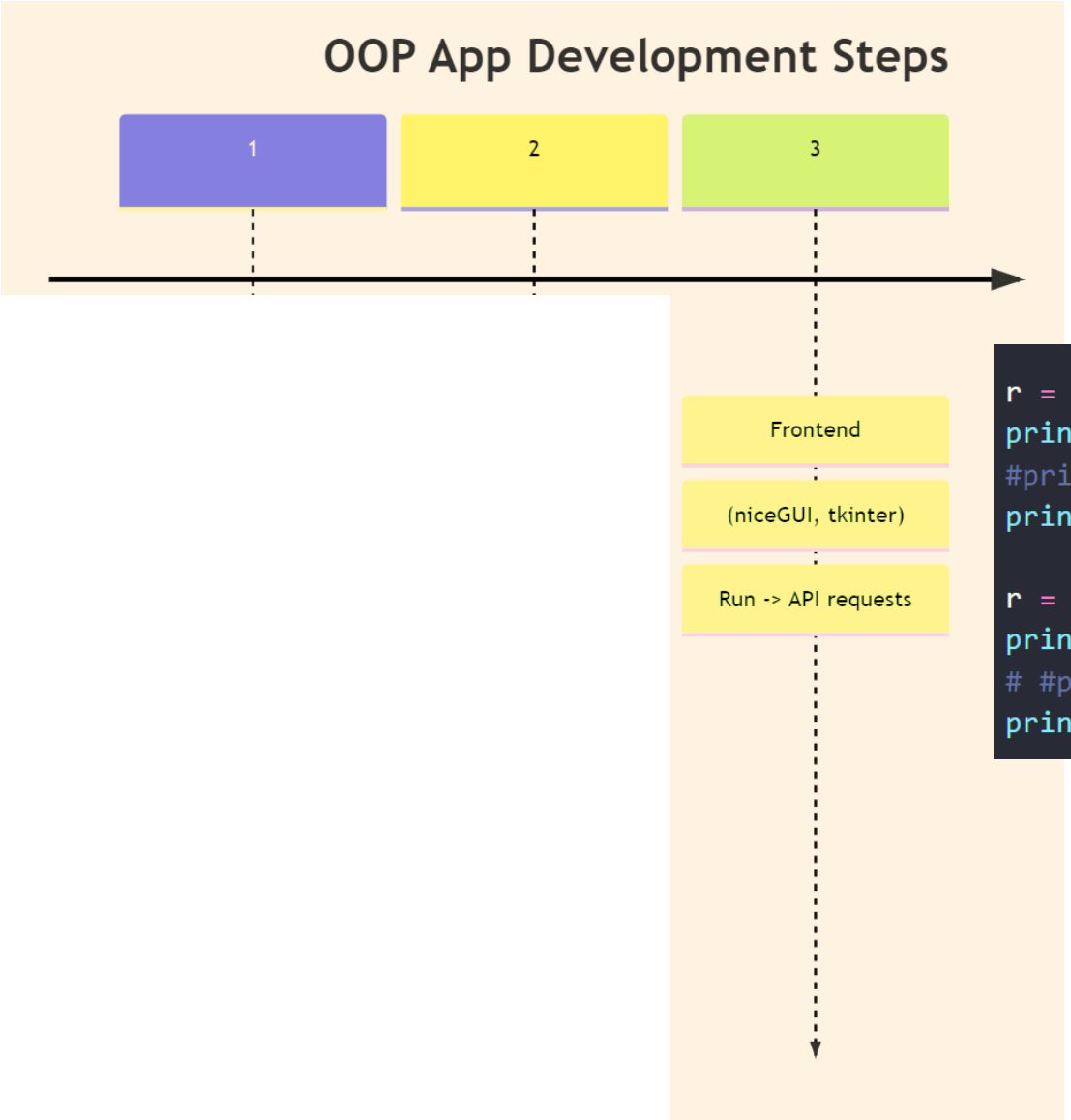
Tkinter GUI : tk_meet.py

Search Button -> on_click()

-> relate to API_ENDPOINT1


```python
def on_click():
    payload = {
        "start_date": st_date.get(),
        "start_time": st_time.get(),
        "end_date": end_date.get(),
        "end_time": end_time.get(),
        "capacity": capacity.get()
    }
    response = requests.post(API_ENDPOINT1, json=payload)
    if response.ok:
        data = response.json()
        row_count = 6
        data = data['Data']
        # for key, value in data.items():
        #     room=key+" : "+str(value)
        #     Label(root, text=room).grid(row=row_count, column=0, padx=10, ipady=5, sticky='E')
        #     row_count +=1
        om = OptionMenu(root, select_opt, *data)
        om.grid(row=row_count, column=0)
        om.config(width=15)
```

OOP App Development Steps



book_room_request.py

```
r = requests.post("http://127.0.0.1:8000/book_room",data=json.dumps(meet_info))
print(r)
#print(dir(r))
print(r.json())

r = requests.get("http://127.0.0.1:8000/book_room")
print(r)
# #print(dir(r))
print(r.json())
```

Run on New Terminal
-> python book_room_request.py