Name: Sonal Bedi

Employee Id: 22704

Program: Software Developer (Framework Engineer) Python Training

Python Training

Project-6

Title: Create Movie Booking System using FastApi Concept.

In this project, we will discuss how to develop a **movie booking system and how to store** data in database.As in this we our using Mysql database.and also all the operations i.e to add
,update,display,and delete(CRUD) data from database Mysql using restapi.
Most of the functionalities we will be developing in this project :

Table 1:Users

user_id(pk),name,email

Table 2:Movie

Movie_id(pk),movie_name

,movie_description,rating,type

Table 3:Shows

Show_id(pk),movie_id(fk),timeslot_id,the

atre_id,seats_booked,seat_available

Table 4:Theatre

theatre_id(pk),theatre_

name

Table 5:Seats

Seat_id(pk),seat_type,s

eat_price

Table 6:Ticket_Booked

ticket_id(pk),show_id(fk), seat_id(fk), theatre_id(fk)

user_id(fk),timeslot_id(fk),movie_id

- Get:Display all records
- Get/id: Display details of record by id
- Post/:Add new record
- Put/id: Update details of record given id
- Delete/id: Delete given record from database with given id

PURPOSE :

Some important goals of movie ticket booking are as follows:

• Online movie ticket booking system project is aimed to provide facility to book movie tickets online anytime and from anywhere without going to Theatres.

• It involves less number of staffs at the ticket-boxwindow.

• Promote new movies over the internet and gain maximum profit.

• Provide a 24x7 service to the customer.

• Faster reliable sytem.

• The main objective of the Movie Ticket Booking System is to manage the details of Seats, Booking, Customer, Payment, Shows. It manages all the information about Seats, Movie, Shows, Seats.

TOOLS AND TECHNOLOGIES USED:

This project is completely developed using FastAPI for making the APIs for performing all the CRUD operations on the database ,the database used for this project is MySQL and used the SQLALCHEMY ORM and PYDANTIC libraries. For the API testing purposes, used the SWAGGER UI.

THE PROBLEM STATEMENT :

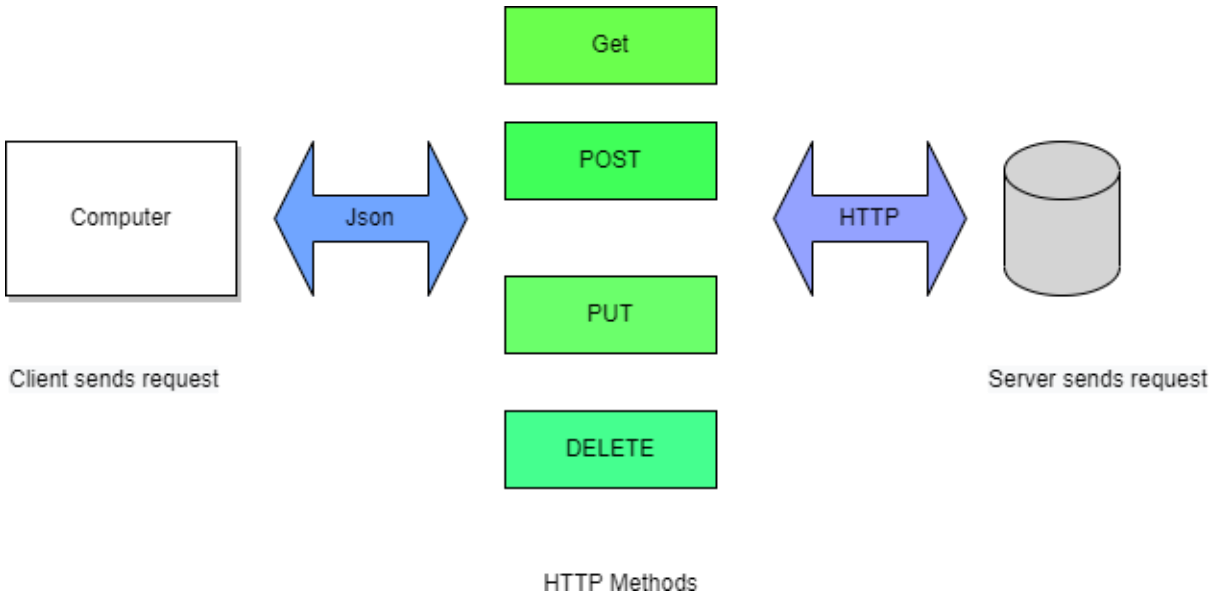
This project requires to make APIs for the steps involved in booking a movie ticket where one has to take care of every situation that may arise during booking a movie ticket. This project also requires one to handle all the corner cases and errors that can occur during the flow of this project.

FLOW OF THE PROJECT:

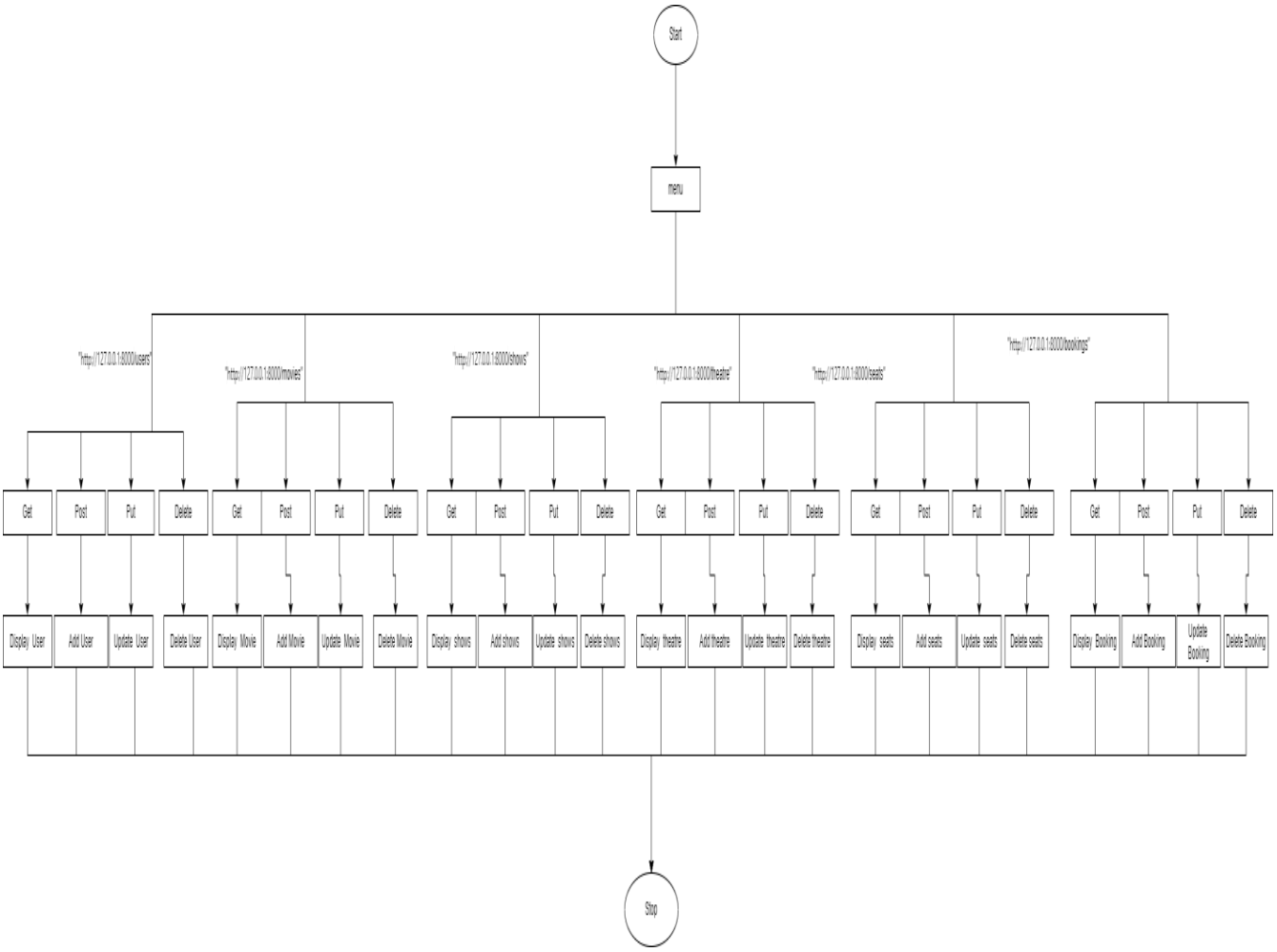
This project has APIs made specifically for multiple steps involved during booking a movie ticket. Specifically this project has six dedicated APIs for :

- Adding a new movie
- Booking a movie ticket
- Fetching the booking details of a user
- Listing all the available movies
- Fetching the total number of available seats for a particular movie
- The First three APIs mentioned in the above list are for generating a POST request and the rest three APIs for generating the GET request.

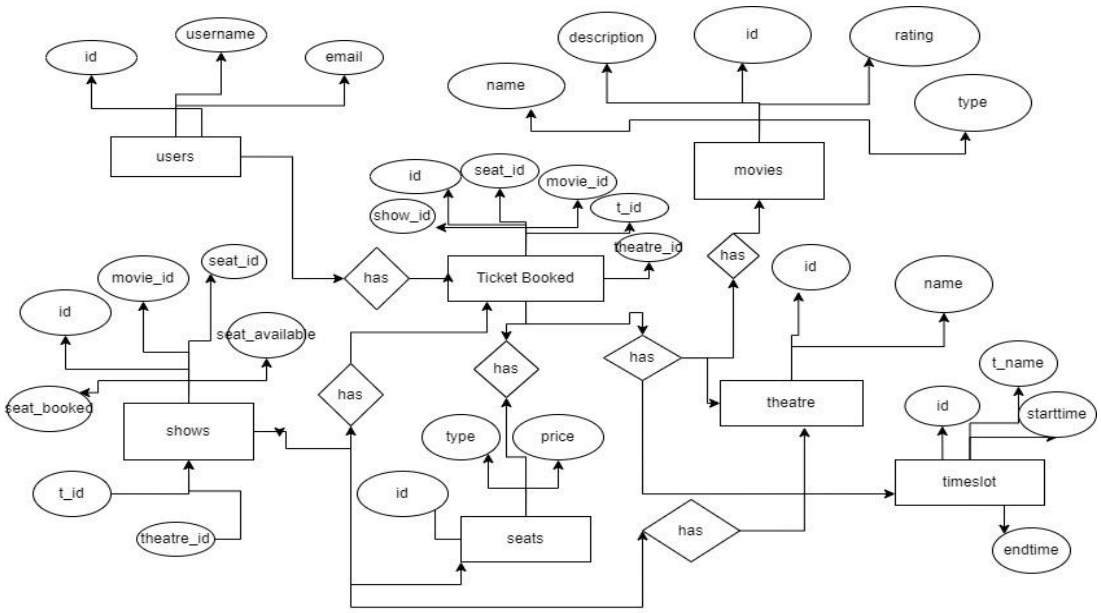
Architecture Diagram:



Flow Diagram:



ER-Diagram:



1. Install FastAPI

   The first step is to install FastAPI

   pip install fastapi

2. Install uvicorn to work as the server:

   pip install uvicorn

3. Install database requirements

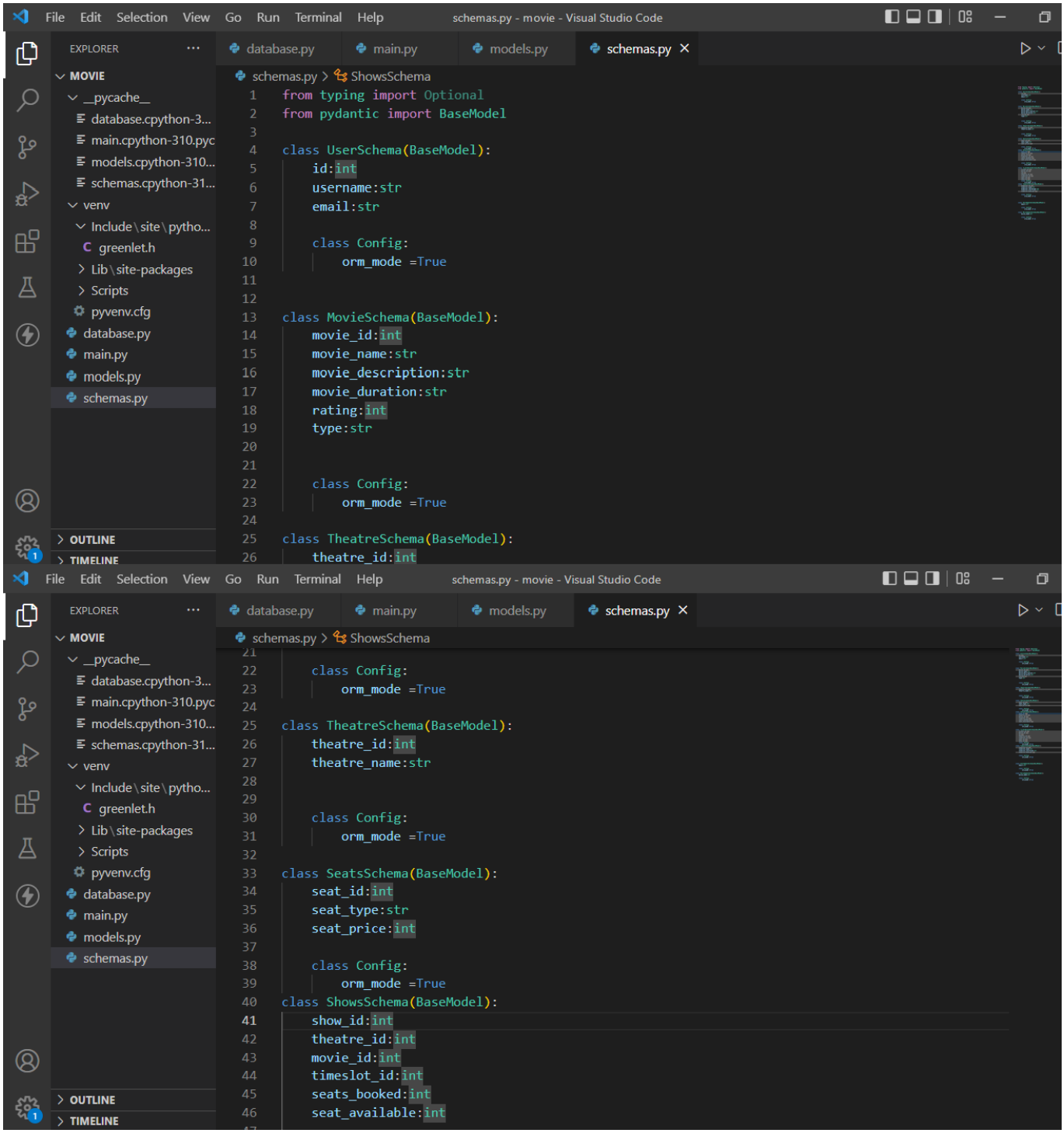
   pip install fastapi, uvicorn**,**pip install sqlalchemy

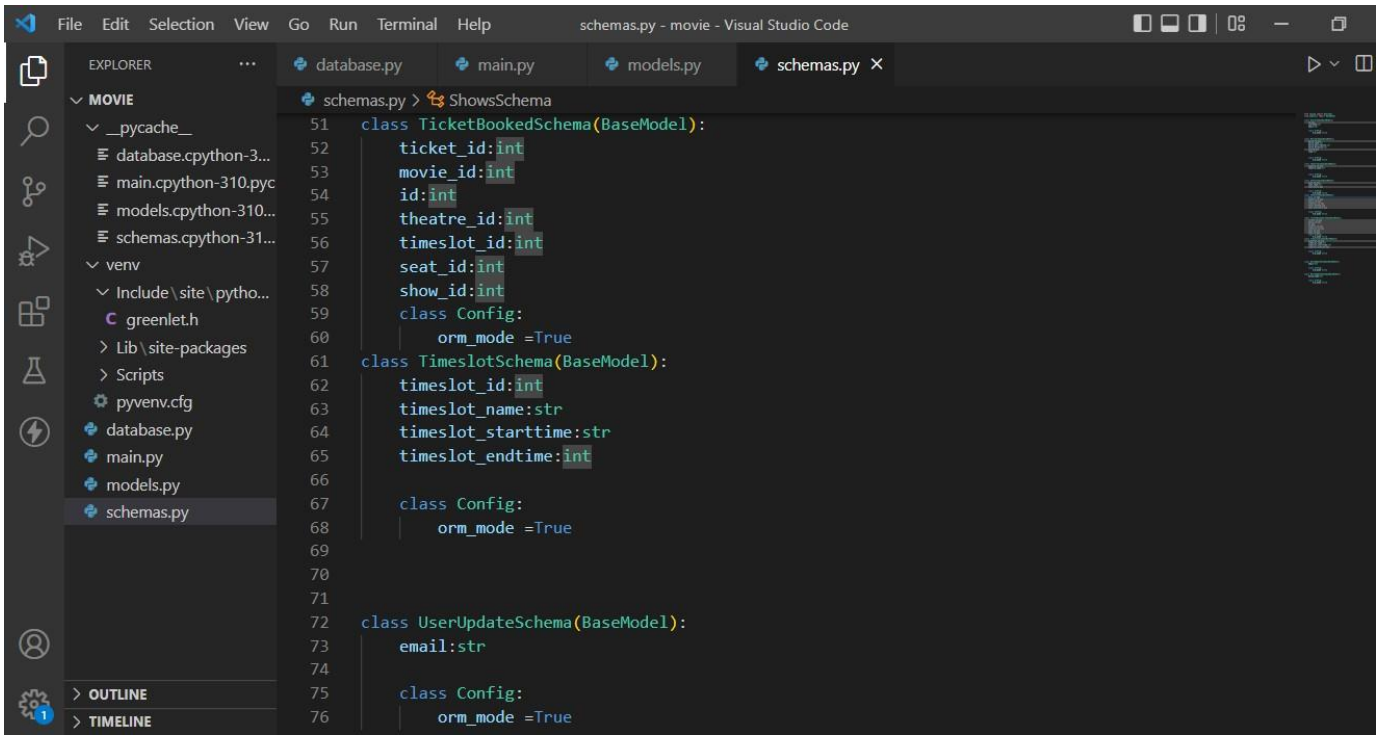
4. To Run:

   Uvicorn main:app --reload

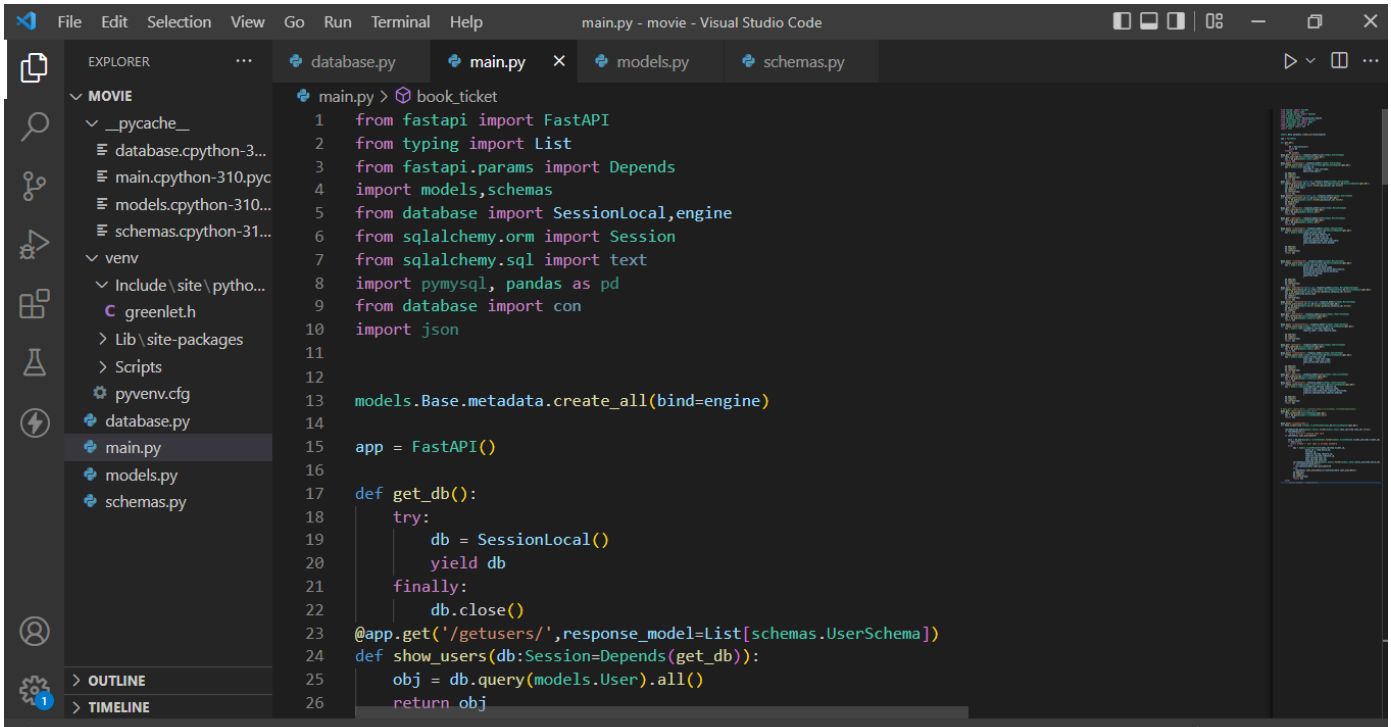
5.db.py

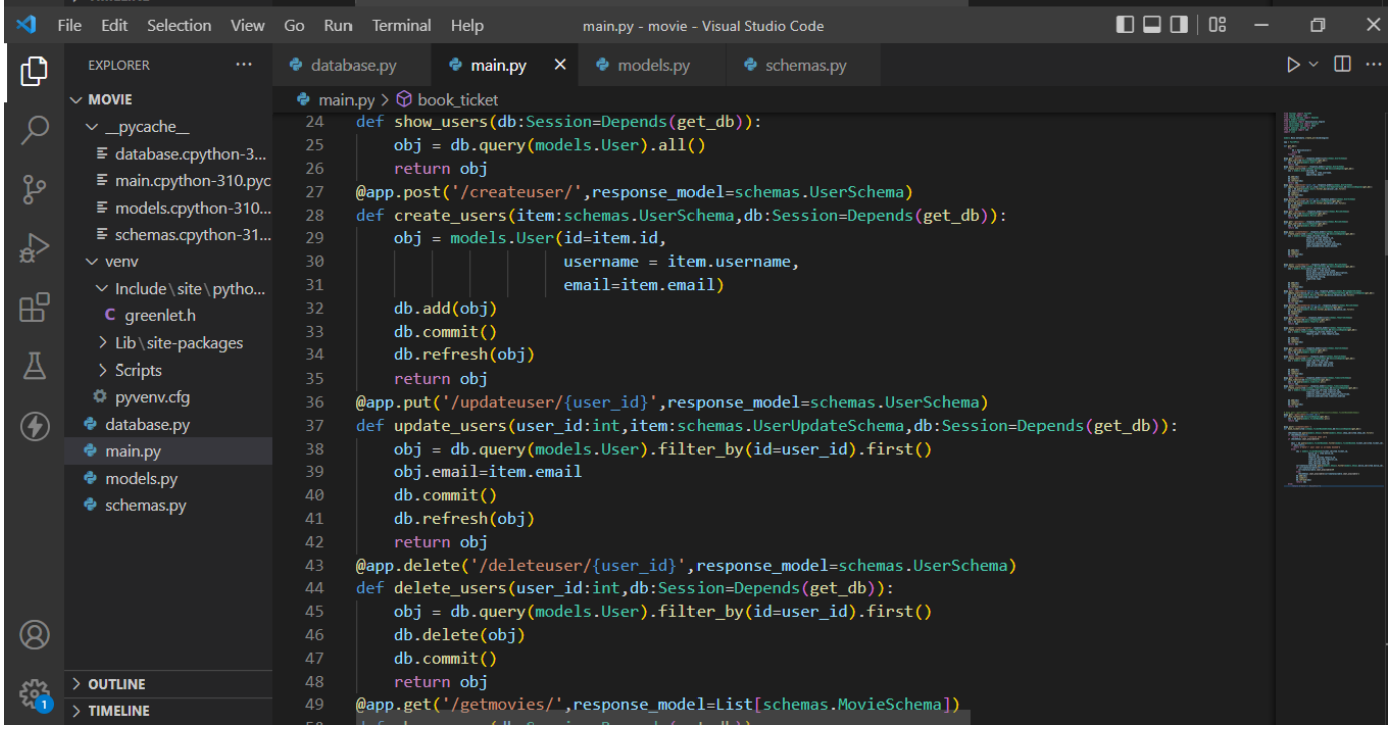
1.schema:



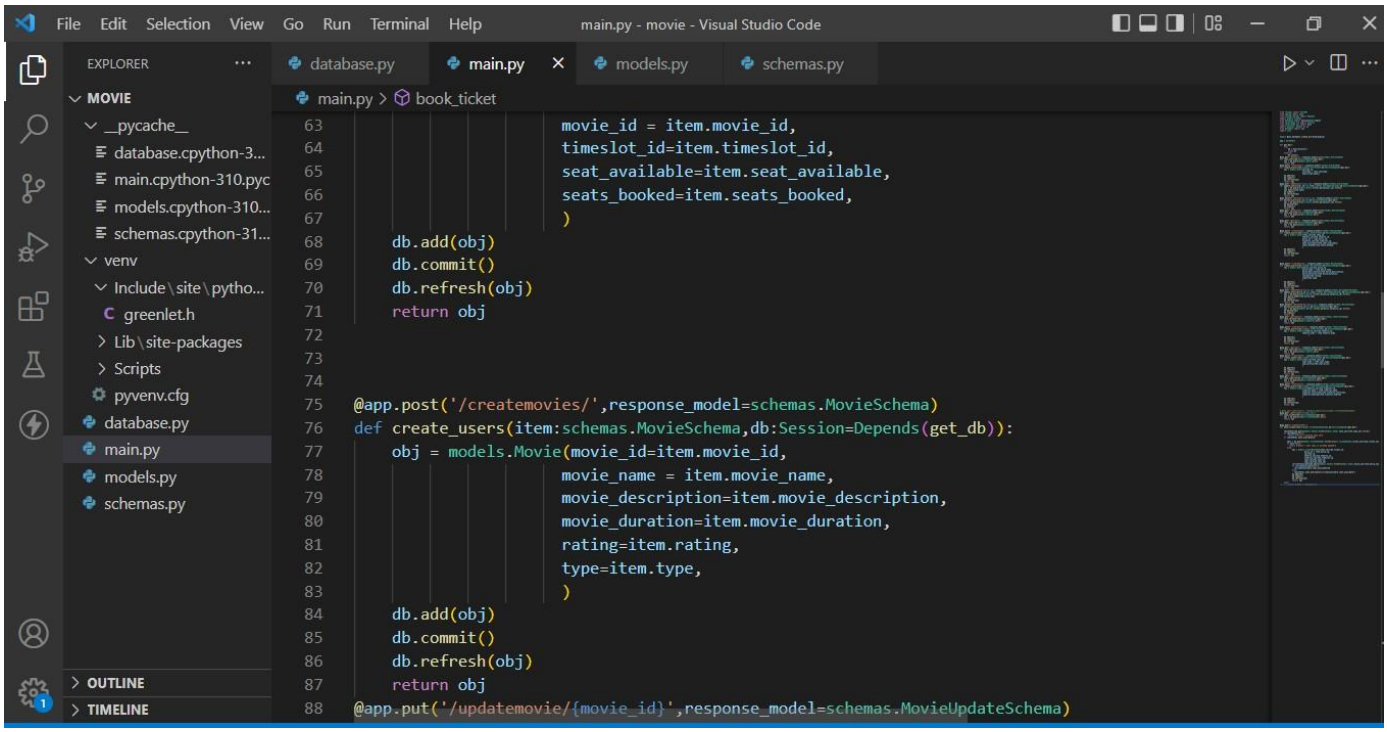

File Edit Selection View Go Run Terminal Help
schemas.py - movie - Visual Studio Code
EXPLORER
MOVIE
__pycache__
database.cpython-3...
main.cpython-310.pyc
models.cpython-310...
schemas.cpython-31...
venv
Include\site\pytho...
greenlet.h
Lib\site-packages
Scripts
pyvenv.cfg
database.py
main.py
models.py
schemas.py
OUTLINE
TIMELINE
database.py
main.py
models.py
schemas.py
schemas.py > ShowsSchema
class TicketBookedSchema(BaseModel):
ticket_id:int
movie_id:int
id:int
theatre_id:int
timeslot_id:int
seat_id:int
show_id:int
class Config:
orm_mode =True
class TimeslotSchema(BaseModel):
timeslot_id:int
timeslot_name:str
timeslot_starttime:str
timeslot_endtime:int
class Config:
orm_mode =True
class UserUpdateSchema(BaseModel):
email:str
class Config:
orm_mode =True

2. Routes for users:

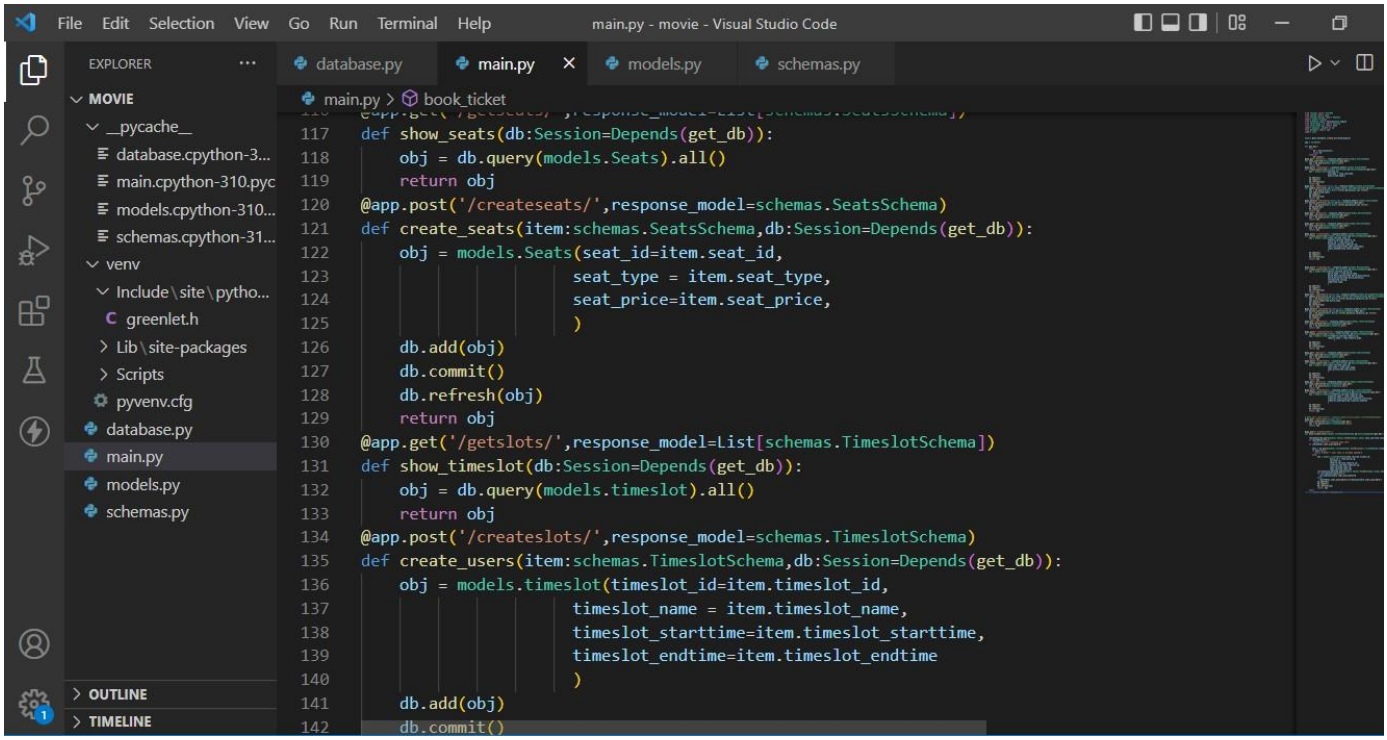




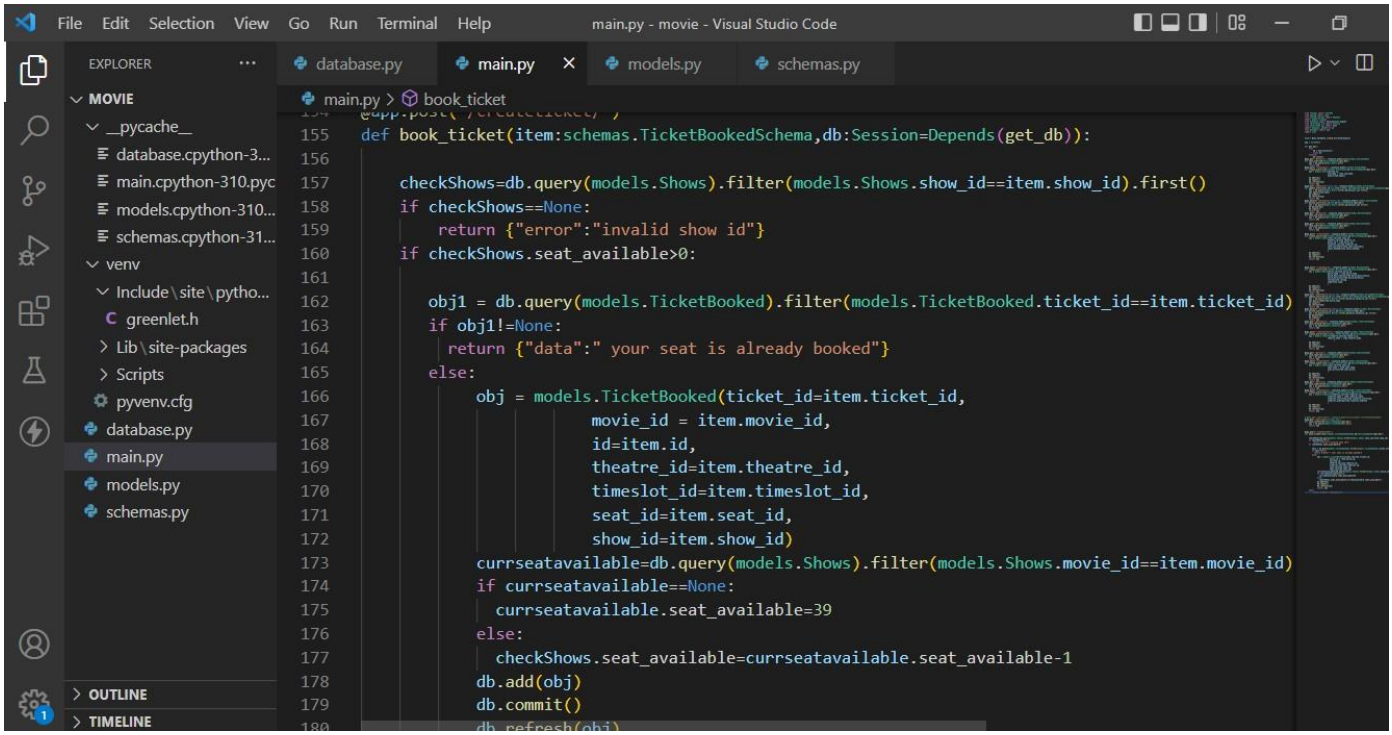
3. Routes for movies:



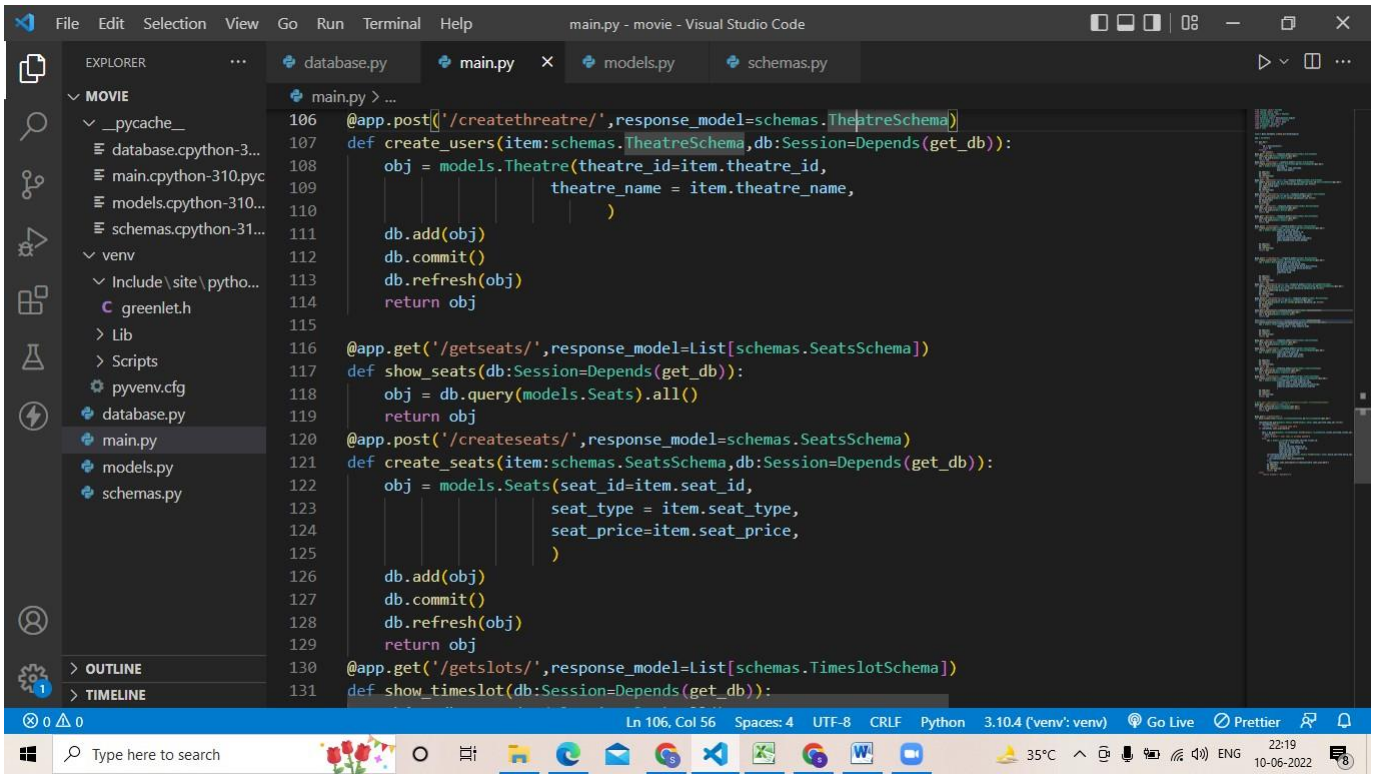
4.Routes for slots:



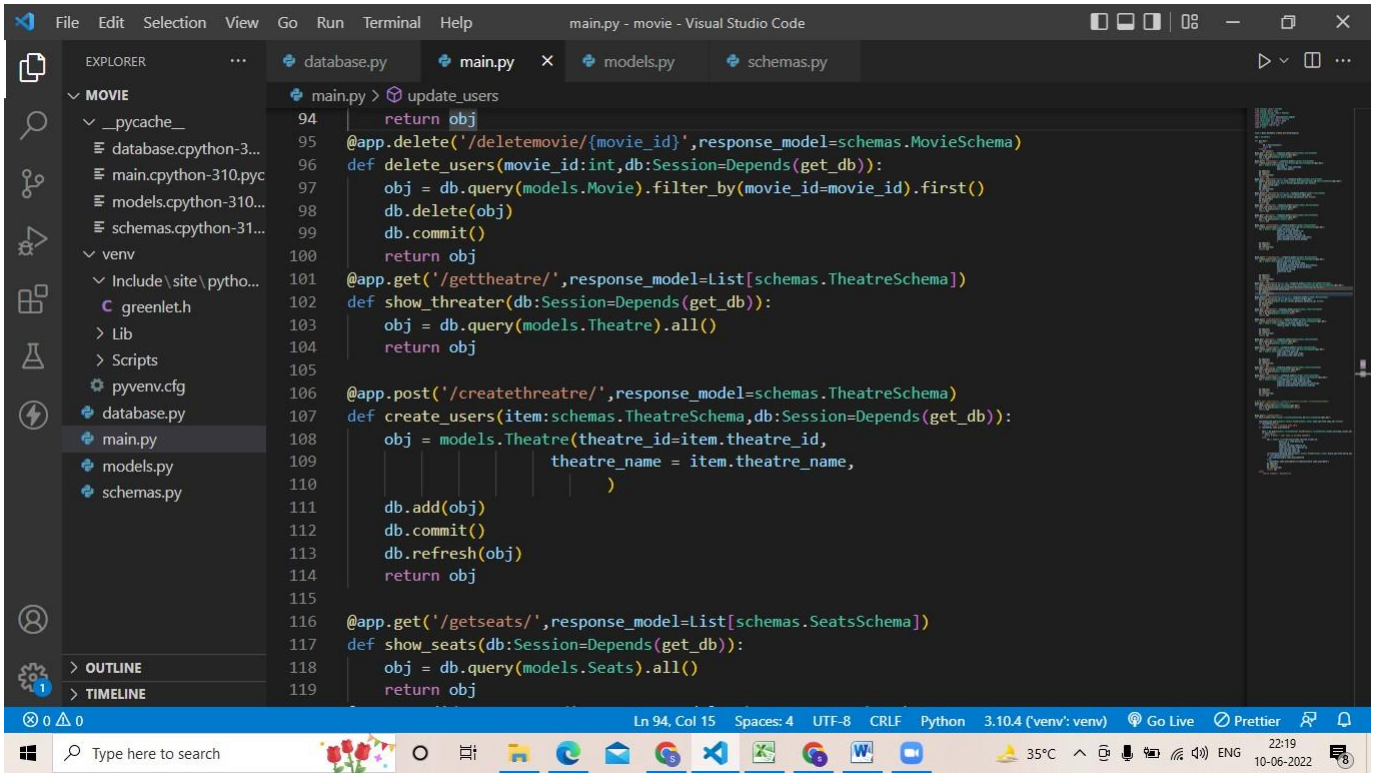
5.ROUTES FOR BOOKING:



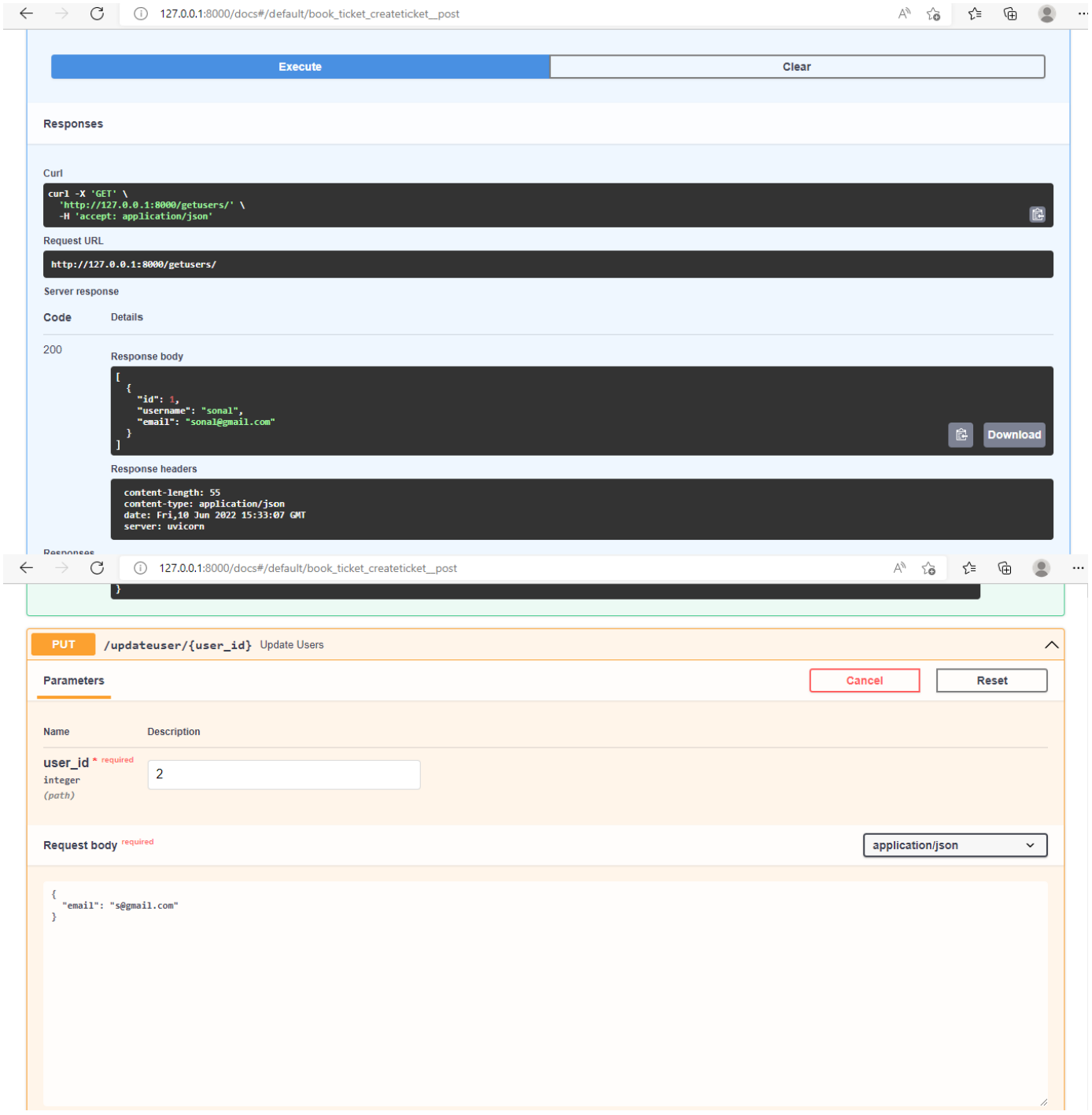
6. Routes for Seats :



7. Routes for Theatre:



OUTPUT:


127.0.0.1:8000/docs#/default/book_ticket_createticket__post
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/getusers/' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/getusers/
Server response
Code
Details
200
Response body
[
{
"id": 1,
"username": "sonal",
"email": "sonal@gmail.com"
}
]
Download
Response headers
content-length: 55
content-type: application/json
date: Fri,10 Jun 2022 15:33:07 GMT
server: uvicorn
127.0.0.1:8000/docs#/default/book_ticket_createticket__post
PUT
/updateuser/{user_id}
Update Users
Parameters
Cancel
Reset
Name
Description
user_id * required
integer
(path)
2
Request body required
application/json
{
"email": "s@gmail.com"
}

127.0.0.1:8000/docs#/default/book_ticket_createticket__post

```
  }
 ]
}
```

DELETE `/deleteuser/{user_id}` Delete Users

Parameters | Cancel

Name	Description
user_id * required integer *(path)*	2

Execute | Clear

Responses

Curl

```
curl -X 'DELETE' \
  'http://127.0.0.1:8000/deleteuser/2' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/deleteuser/2
```

127.0.0.1:8000/docs#/default/show_users_getmovies__get

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/getmovies/' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/getmovies/
```

Server response

Code	Details
200	Response body

```
[
  {
    "movie_id": 101,
    "movie_name": "shiddat2",
    "movie_description": "nice",
    "movie_duration": "120mins",
    "rating": 5,
    "type": "emotional"
  }
]
```

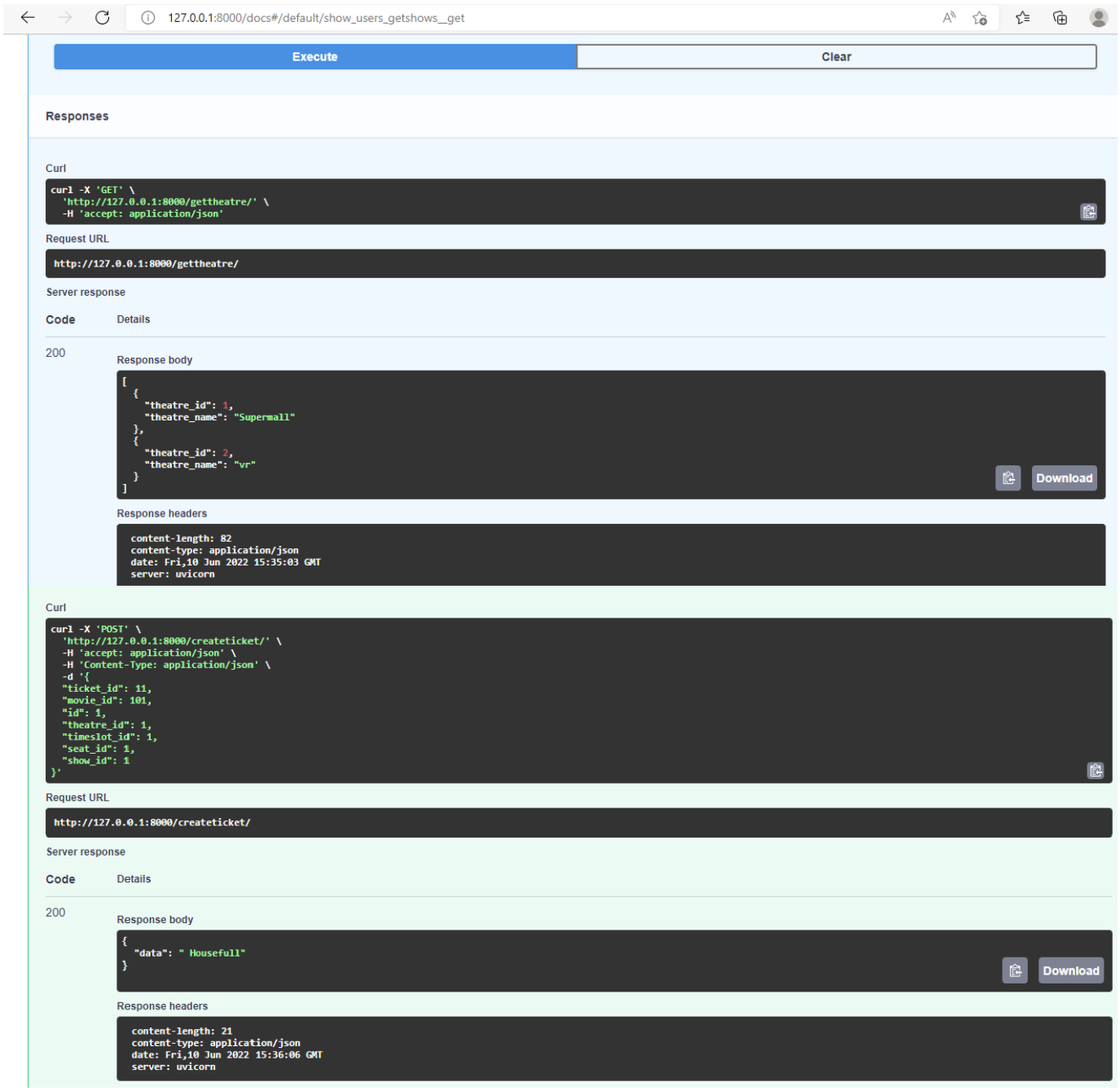
Download

Response headers

```
content-length: 126
content-type: application/json
date: Fri,10 Jun 2022 15:34:27 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*


127.0.0.1:8000/docs#/default/show_users_getshows__get
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://127.0.0.1:8000/gettheatre/' \
-H 'accept: application/json'
Request URL
http://127.0.0.1:8000/gettheatre/
Server response
Code
Details
200
Response body
[
{
"theatre_id": 1,
"theatre_name": "Supermall"
},
{
"theatre_id": 2,
"theatre_name": "vr"
}
]
Download
Response headers
content-length: 82
content-type: application/json
date: Fri,10 Jun 2022 15:35:03 GMT
server: uvicorn
Curl
curl -X 'POST' \
'http://127.0.0.1:8000/createticket/' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"ticket_id": 11,
"movie_id": 101,
"id": 1,
"theatre_id": 1,
"timeslot_id": 1,
"seat_id": 1,
"show_id": 1
}'
Request URL
http://127.0.0.1:8000/createticket/
Server response
Code
Details
200
Response body
{
"data": " Housefull"
}
Download
Response headers
content-length: 21
content-type: application/json
date: Fri,10 Jun 2022 15:36:06 GMT
server: uvicorn

```
  "ticket_id": 1,
  "movie_id": 101,
  "id": 1,
  "theatre_id": 1,
  "timeslot_id": 1,
  "seat_id": 1,
  "show_id": 1
}'
```

Request URL

```
http://127.0.0.1:8000/createticket/
```

Server response

Code	Details
200	Response body

```
{
  "data": " Housefull"
}
```

Download

Response headers

```
content-length: 21
content-type: application/json
date: Fri,10 Jun 2022 15:36:22 GMT
server: uvicorn
```

Responses

Code	Description	Links
200	Successful Response	*No links*

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/createticket/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "ticket_id": 1,
  "movie_id": 101,
  "id": 1,
  "theatre_id": 1,
  "timeslot_id": 1,
  "seat_id": 1,
  "show_id": 55
}'
```

Request URL

```
http://127.0.0.1:8000/createticket/
```

Server response

Code	Details
200	Response body

```
{
  "error": "invalid show id"
}
```

Download