Django Database Part 2

Working on Python command line

Python manage.py shell

اسم التطبيق متعي

>>>from flights.models import Flight

اسم الكلاس اللي عرفته في صفحة models.py

>>>flights=Flight.objects.all()

هني خزنت المعلومات اللي في الجدول في المتغير فلايتس

>>>flights

بعدين ناديت المتغير هدا

This will display the available records in flights object.

Flight: 1: New York to London

.first

>>> flight=Flights.objecs.firs() This command will get the first record in Flights object

And we can access the properties of this object as per the following commands

>>> flight.id (to get the id of the flight in the first record)

1

>>> flight.origin (to get the origin city of the first record)

New York

>>> flight. Destination

London

>>> flight.duration

415

بعد مادرتها مسحلي كل شيء موجود كانو عندي زوز ريكوردس وتوا فضت خلاص

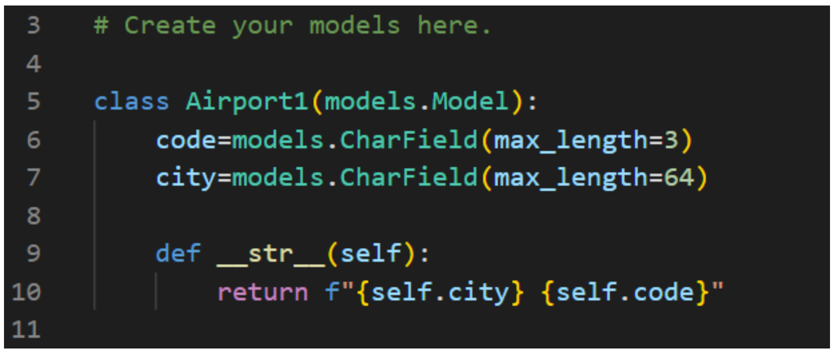
>>>flight.delete() (by executing this command, the first record in the flight variable will be deleted)

Part II of the Flight application

إنشاء جدول مطار لربط أي رحلة بمطار معين

1. Creating Airport table to link any flight to a specific airport

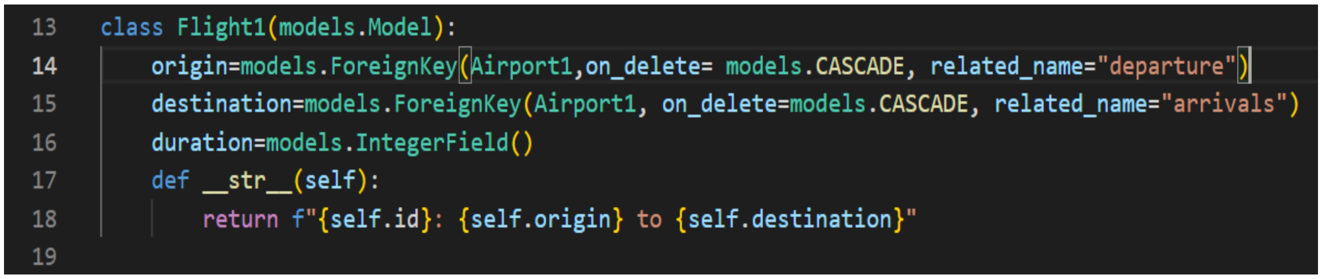
In models.py file in the flights application write the following code in order to create the airport class which will represent the airport table:



```python
# Create your models here.

class Airport1(models.Model):
    code=models.CharField(max_length=3)
    city=models.CharField(max_length=64)

    def __str__(self):
        return f"{self.city} {self.code}"

```

- Where code an attribute that represents the airport code
- City represents the attribute of the airport city
- Def__str__(self): the function that give the string representation for the class object.

2. And change the Flights table accordingly:



```
class Flight1(models.Model):
    origin=models.ForeignKey(Airport1,on_delete= models.CASCADE, related_name="departure")
    destination=models.ForeignKey(Airport1, on_delete=models.CASCADE, related_name="arrivals")
    duration=models.IntegerField()
    def __str__(self):
        return f"{self.id}: {self.origin} to {self.destination}"
```

Where Origin field changed to be a foreign key attribute that holds the airport object as its value,

لحذف سجل الرحلة بمجرد حذف سجل المطار

On_delete-models.CASCADE: to delete the Flight record once the Airport record is been deleted

And the related_name="departure" means: to get all the flights which are departed from a certain airport.

للحصول على جميع الرحلات المغادرة من مطار معين

Destination field changed in the same way as the origin field

لسرد جميع الرحلات التي هبطت في مطار معين

Related_name="arrivals: to list all the flights which landed in a certain airport.

★ To change database in Django we have to apply the two following commands:

- Python manage.py makemigrations
- Python manage.py migrate

Working on Python command line

Python manage.py shell

>>>from flights.models import * → الكلاسات كلهم يعني

By running this command, Django will bring all the tables created for the flights application which are: Airport table and Flights table.

To create new record in Airport table, we run the following python command:

متغير نخزنوا فيه في االريكورد ←

```
>>>jfk=Airport(code="JFK", city="New York")

>>>jfk.save()

>>>lhr=Airport(code="LHR", city="London")

>>>lhr.save()

>>>cdg=Airport(code="CDG", city="Paris")

>>>cdg.save()
```

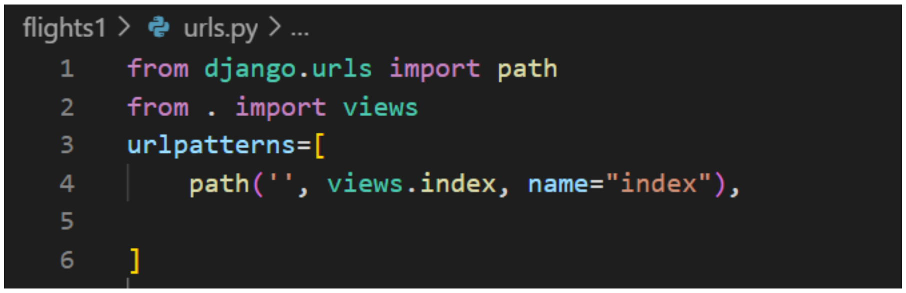
```
>>>nrt=Airport(code="NRT", city="Tokyo")
>>>nrt.save()
```

The following code will add a new flight record based on the above added airport records:

```
>>>f=Flight(origin=jfk, destination=lhr, duration=415)
>>>f.save()
>>>f.origin (by running this command will display the following)
Airport: New York   JFK
>>>lhr.arrivals.all() (this command will list all the flights arrived at the airport LHR)
```

Creating a display page to list all the flights

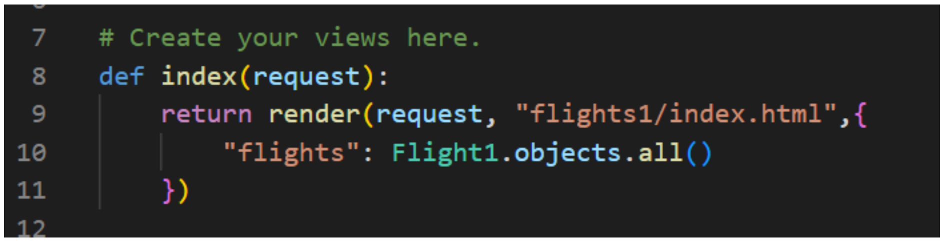
1. urls.py file:



flights1 > urls.py > ...

```python
from django.urls import path
from . import views
urlpatterns=[
    path('', views.index, name="index"),

]
```

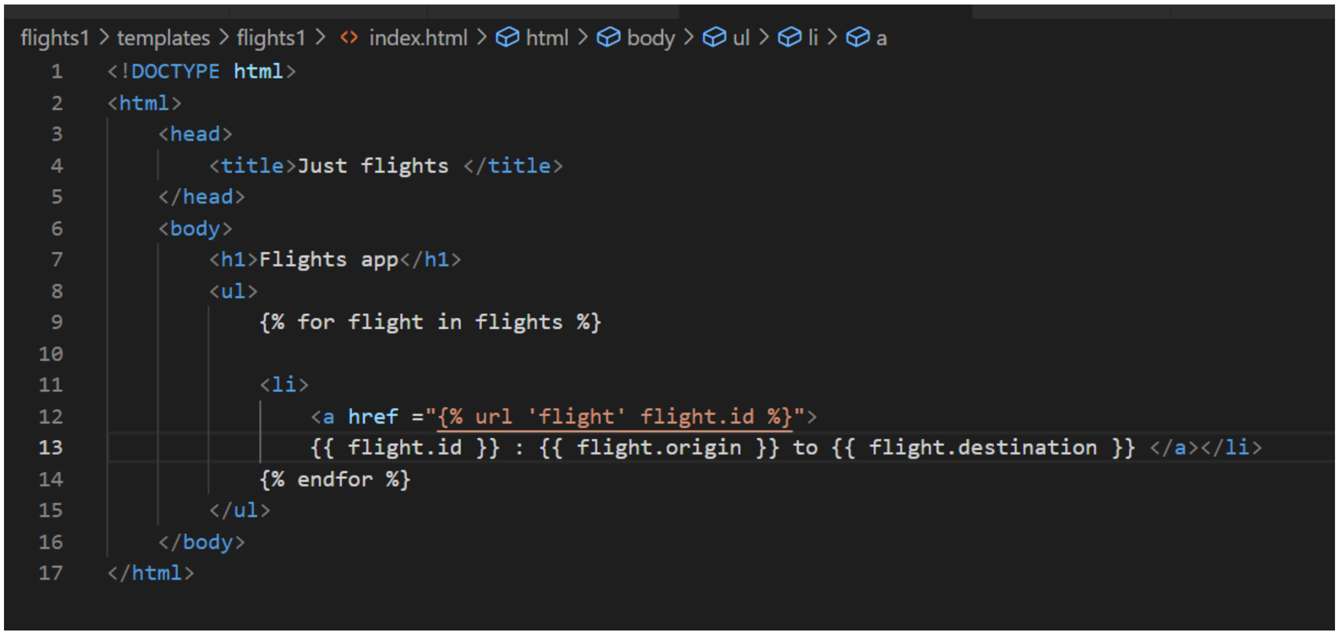
2. Views.py (to create the index view function:



```python
# Create your views here.
def index(request):
    return render(request, "flights1/index.html",{
        "flights": Flight1.objects.all()
    })
```

This view will render the index.html page that is going to display all flight records.

3. templates->flights->index.html file:

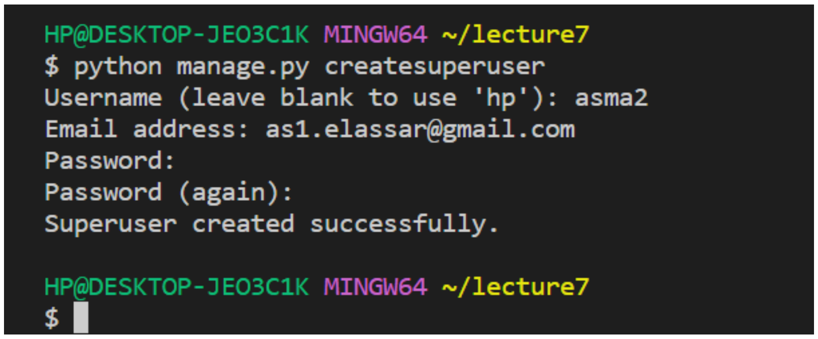

flights1 > templates > flights1 > index.html > html > body > ul > li > a

```
<!DOCTYPE html>
<html>
    <head>
        <title>Just flights </title>
    </head>
    <body>
        <h1>Flights app</h1>
        <ul>
            {% for flight in flights %}

            <li>
                <a href ="{% url 'flight' flight.id %}">
                {{ flight.id }} : {{ flight.origin }} to {{ flight.destination }} </a></li>
            {% endfor %}
        </ul>
    </body>
</html>
```

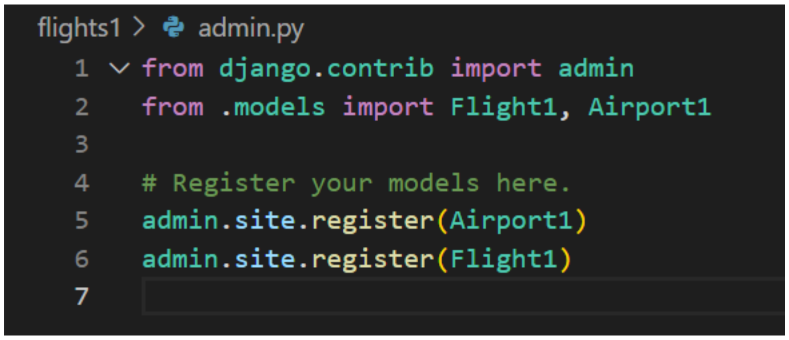
Manipulating database via the Django admin web interface

1. create a super user account in order to access admin web interface by running the following commands:

Python manage.py createsuperuser


```
HP@DESKTOP-JEO3C1K MINGW64 ~/lecture7
$ python manage.py createsuperuser
Username (leave blank to use 'hp'): asma2
Email address: as1.elassar@gmail.com
Password:
Password (again):
Superuser created successfully.

HP@DESKTOP-JEO3C1K MINGW64 ~/lecture7
$
```

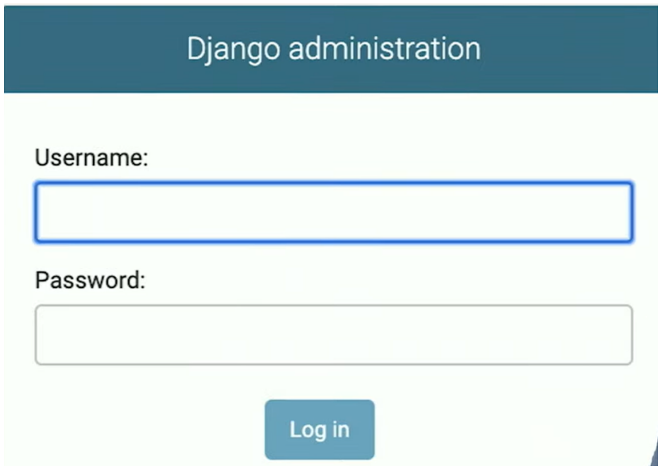
2. go to admin.py file under flights application and change the following:

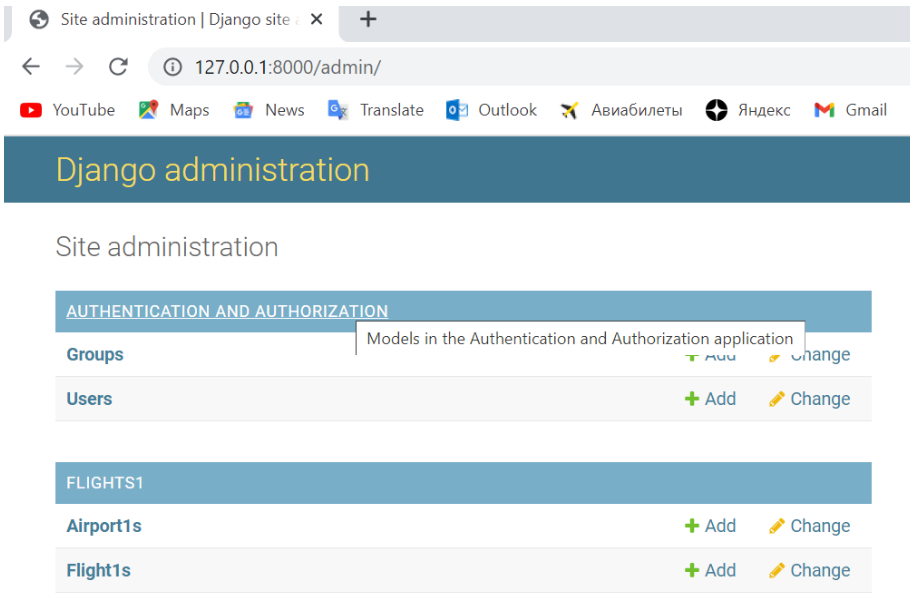


```
flights1 > admin.py
from django.contrib import admin
from .models import Flight1, Airport1

# Register your models here.
admin.site.register(Airport1)
admin.site.register(Flight1)
```

This is going to tell Django admin app to manipulate Airport and Flights tables

By entering the credentials for Django superuser in the following login page





Site administration | Django site

127.0.0.1:8000/admin/

Django administration

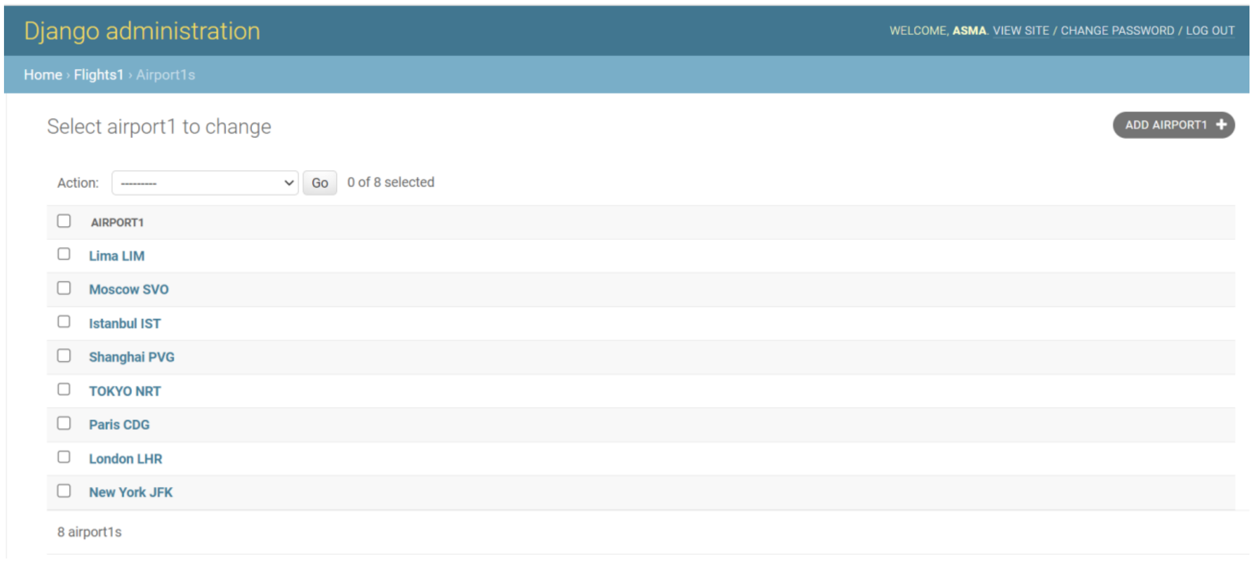
Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

Models in the Authentication and Authorization application

FLIGHTS1		
Airport1s	+ Add	Change
Flight1s	+ Add	Change

By clicking on Airports you can add new airport:



Django administration

WELCOME, ASMA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Flights1 › Airport1s

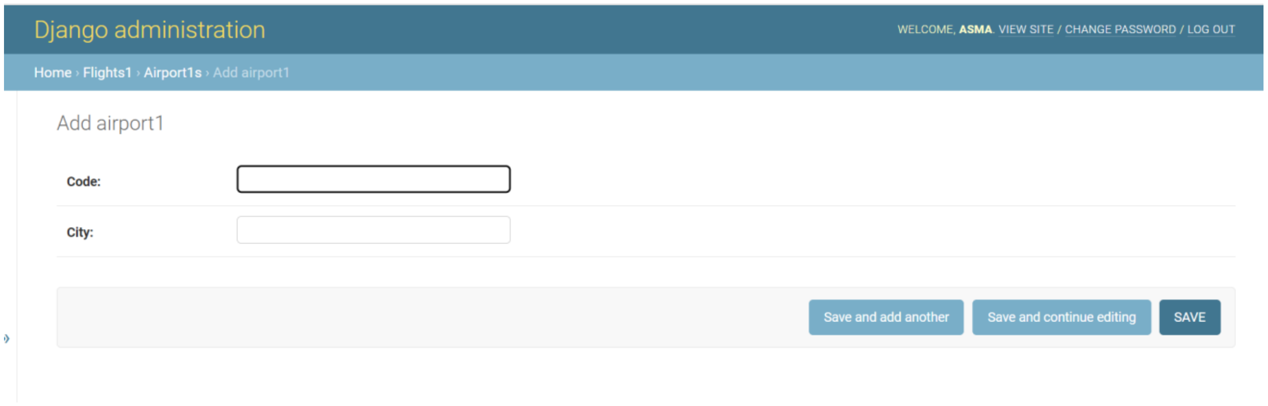
Select airport1 to change

ADD AIRPORT1

Action: --------- Go 0 of 8 selected

AIRPORT1
Lima LIM
Moscow SVO
Istanbul IST
Shanghai PVG
TOKYO NRT
Paris CDG
London LHR
New York JFK

8 airport1s



Django administration

WELCOME, ASMA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Flights1 › Airport1s › Add airport1

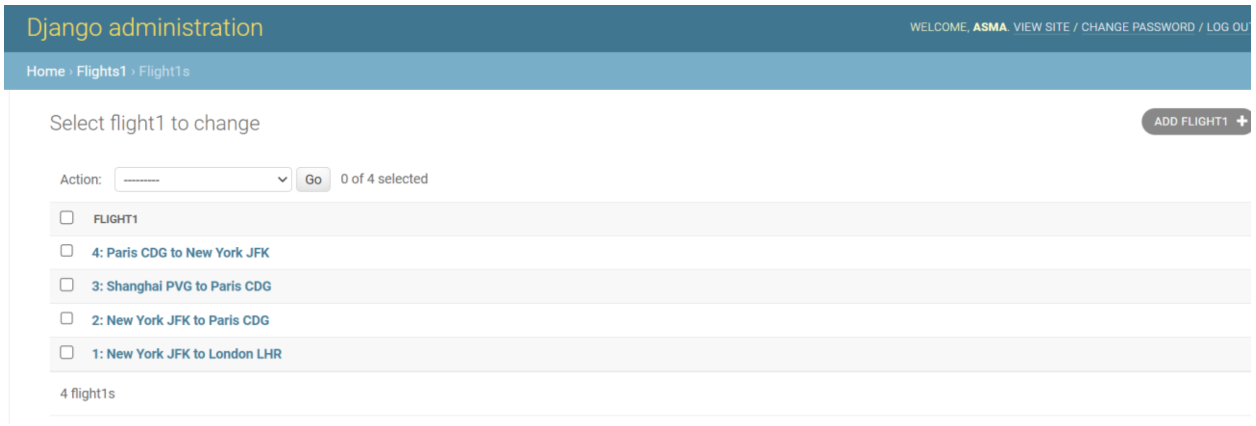
Add airport1

Code:

City:

Save and add another | Save and continue editing | SAVE

By using the same Django admin interface we can add flights:



Django administration

WELCOME, ASMA. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Flights1 › Flight1s

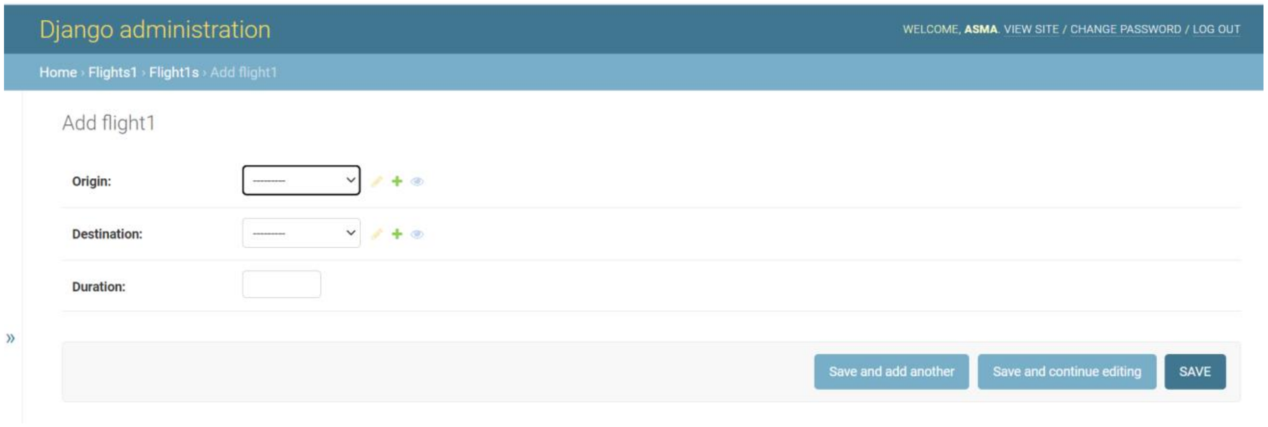
Select flight1 to change

ADD FLIGHT1

Action: --------- Go 0 of 4 selected

FLIGHT1
4: Paris CDG to New York JFK
3: Shanghai PVG to Paris CDG
2: New York JFK to Paris CDG
1: New York JFK to London LHR

4 flight1s


Django administration
WELCOME, ASMA. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Flights1 › Flight1s › Add flight1
Save and add another
Save and continue editing
SAVE

Add flight1

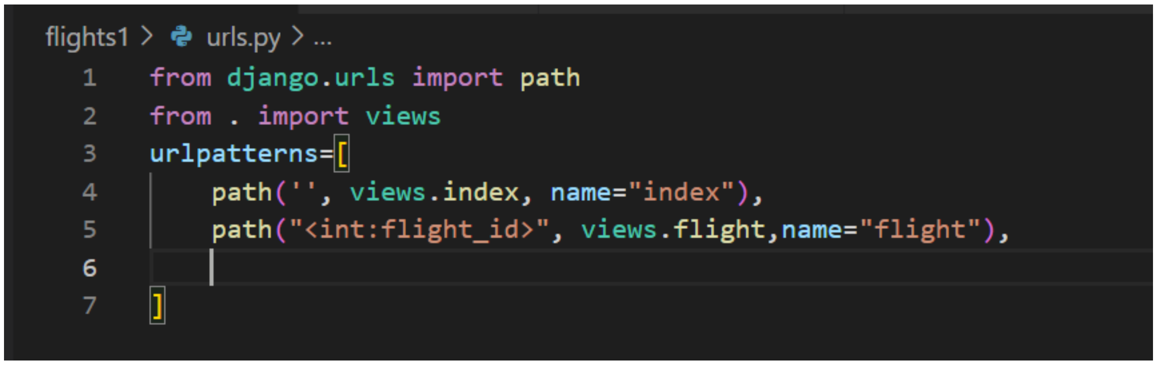
Origin: ---------

Destination: ---------

Duration:

Adding flight page to the flights application that display a specific flight details, when entering a flight number in the url:

- urls.py file:



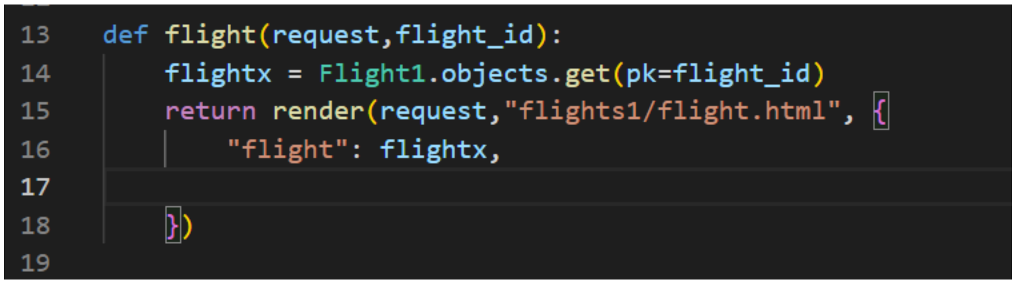
flights1 > urls.py > ...

```python
from django.urls import path
from . import views
urlpatterns=[
    path('', views.index, name="index"),
    path("<int:flight_id>", views.flight,name="flight"),
    
]
```

Where flight_id is an integer variable that will get the flight number from the url entered by the user.

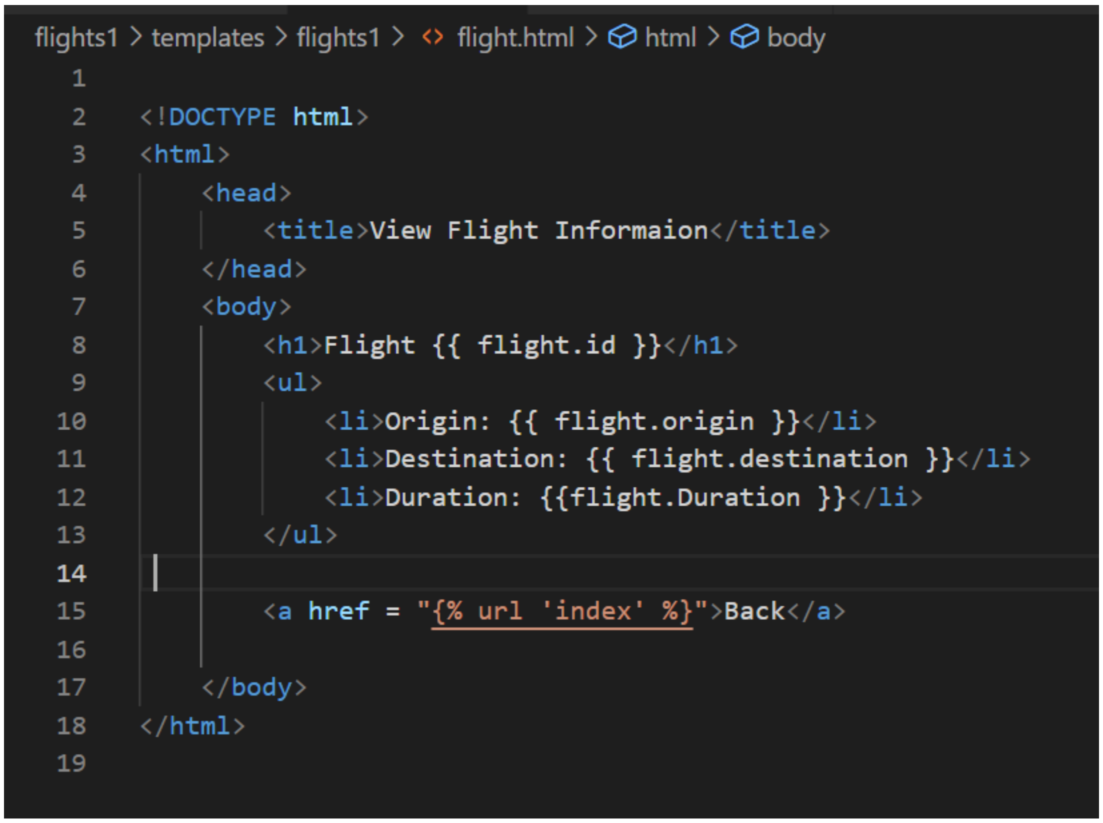
- Views.py

Creating a flight function view that will call the flight.html file , this function has the flight_id returned from the url entered by user, and it will return flightx object which will get flight record which its primary key (pk) is the flight_id entered by the user:



```python
def flight(request,flight_id):
    flightx = Flight1.objects.get(pk=flight_id)
    return render(request,"flights1/flight.html", {
        "flight": flightx,

    })

```

- Templates->flights->flight.html

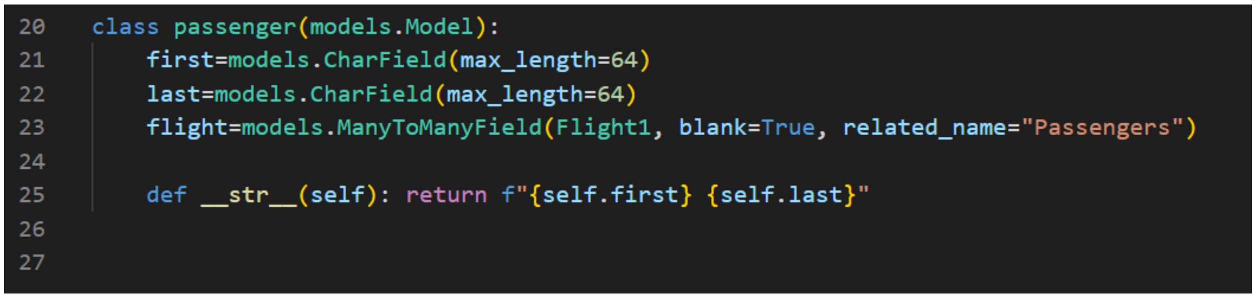


flights1 > templates > flights1 > flight.html > html > body

```html

<!DOCTYPE html>
<html>
    <head>
        <title>View Flight Informaion</title>
    </head>
    <body>
        <h1>Flight {{ flight.id }}</h1>
        <ul>
            <li>Origin: {{ flight.origin }}</li>
            <li>Destination: {{ flight.destination }}</li>
            <li>Duration: {{flight.Duration }}</li>
        </ul>

        <a href = "{% url 'index' %}">Back</a>

    </body>
</html>

```

Adding Passengers to a flight:

1. in models.py create a passenger class to represent passenger's table:



```python
class passenger(models.Model):
    first=models.CharField(max_length=64)
    last=models.CharField(max_length=64)
    flight=models.ManyToManyField(Flight1, blank=True, related_name="Passengers")

    def __str__(self): return f"{self.first} {self.last}"


```

First: represents passenger first name

Last: represents passenger last name

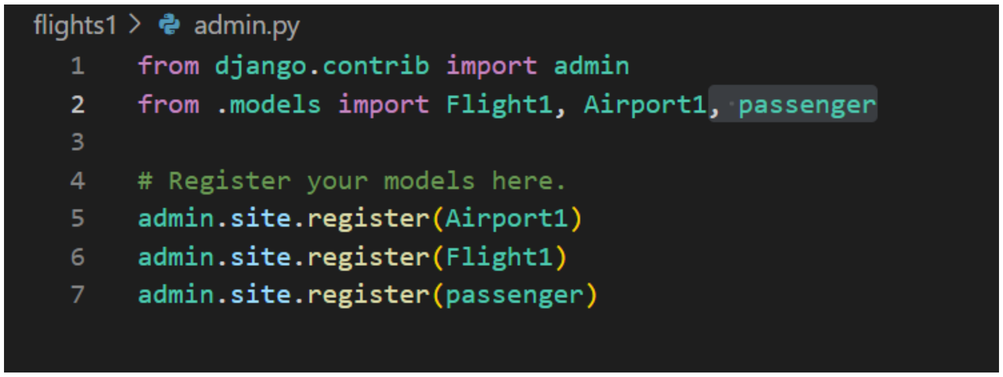
Flight: represents the flight booked by the passenger, تمثل الرحلة التي حجزها الراكب blank: means the passenger can have zero flight booked, ، يعني أن المسافر لا يمكنه حجز أي رحلة and the related_name: means list all the passengers booking for the same flight, يعني سرد جميع الركاب الذين حجزوا لنفس الرحلة NanyToManyField means that the passenger may book several flights and the flight can have several passengers. يعني أنه يجوز للمسافر حجز عدة رحلات ويمكن أن تضم الرحلة عدة ركاب.

2. to apply these changes to Django database, run the following commands:

-python manage.py makemigrations

- python manage.py migrate

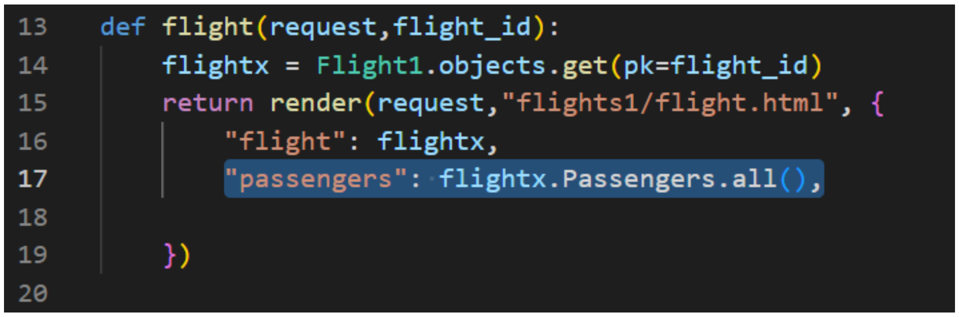
3. In admin.py file, change the following:



flights1 > admin.py

```python
from django.contrib import admin
from .models import Flight1, Airport1, passenger

# Register your models here.
admin.site.register(Airport1)
admin.site.register(Flight1)
admin.site.register(passenger)
```

4. in views.py in flight view function we want to display passengers booking for a particular flight:

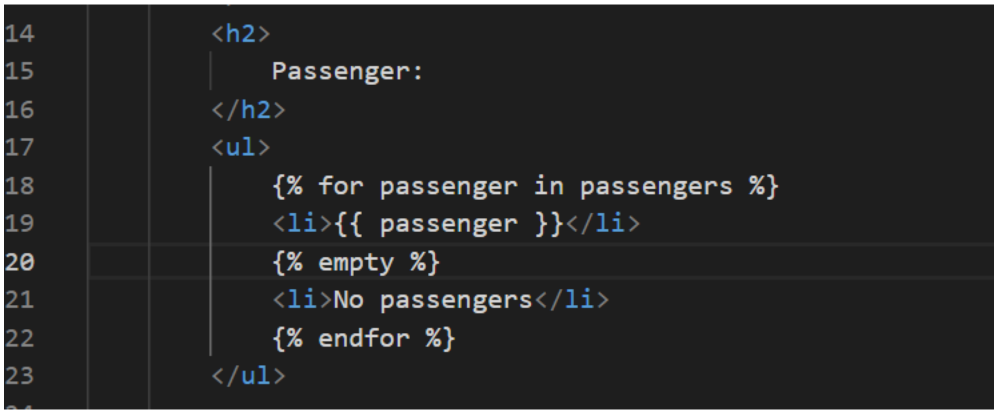


```python
def flight(request,flight_id):
    flightx = Flight1.objects.get(pk=flight_id)
    return render(request,"flights1/flight.html", {
        "flight": flightx,
        "passengers": flightx.Passengers.all(),

    })
```

“Passengers”: flight.passenger.all()

To list all passenger names on a particular flight, Passengers the same name used in the related argument.

5. change the flight.html page accordingly:



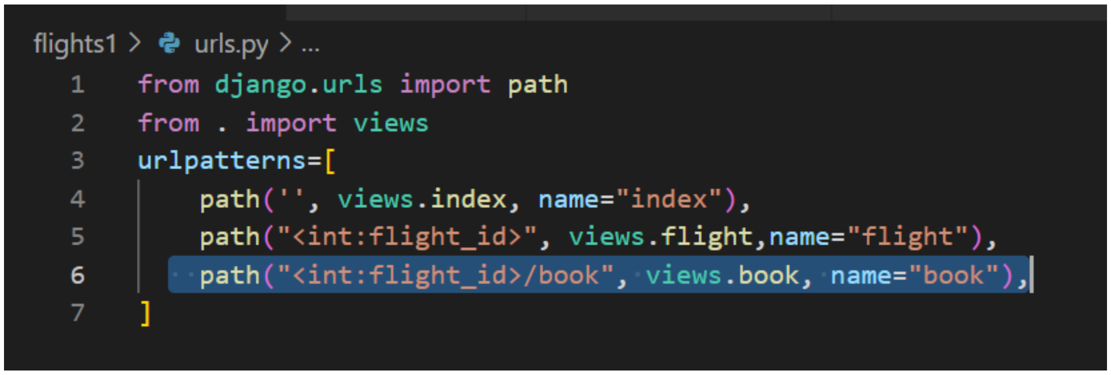
```html
        <h2>
            Passenger:
        </h2>
        <ul>
            {% for passenger in passengers %}
            <li>{{ passenger }}</li>
            {% empty %}
            <li>No passengers</li>
            {% endfor %}
        </ul>
```

{% empty %} means in the case of no passengers booked for that flight.

6. adding new passengers to flight

Now making a new route to book a flight

- In urls.py add the following code:

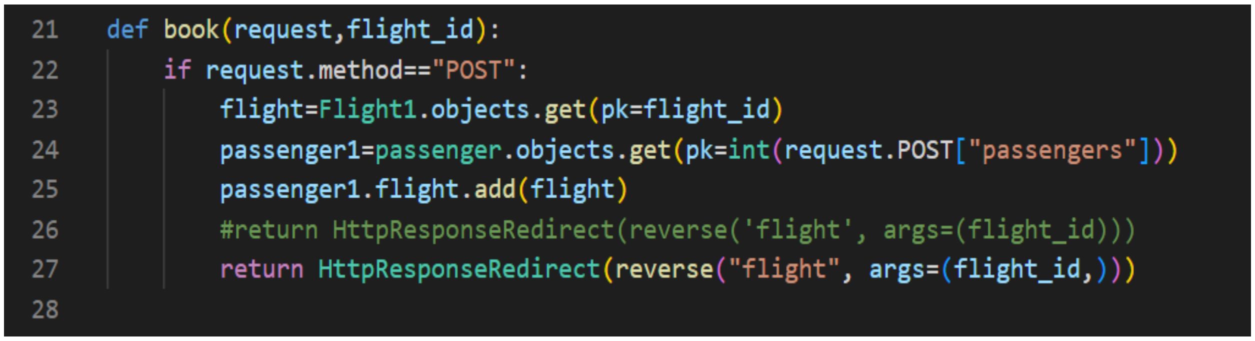


```
flights1 > urls.py > ...
from django.urls import path
from . import views
urlpatterns=[
    path('', views.index, name="index"),
    path("<int:flight_id>", views.flight,name="flight"),
    path("<int:flight_id>/book", views.book, name="book"),
]
```

Where flight_id is the id for a particular flight entered by user in the url

Book is the name of the function view that will perform adding passenger information record to passengers table. هو اسم عرض الوظيفة الذي سيؤدي إلى إضافة سجل معلومات الركاب إلى جدول الركاب

- Views.py



```
def book(request,flight_id):
    if request.method=="POST":
        flight=Flight1.objects.get(pk=flight_id)
        passenger1=passenger.objects.get(pk=int(request.POST["passengers"]))
        passenger1.flight.add(flight)
        #return HttpResponseRedirect(reverse('flight', args=(flight_id)))
        return HttpResponseRedirect(reverse("flight", args=(flight_id,)))
```

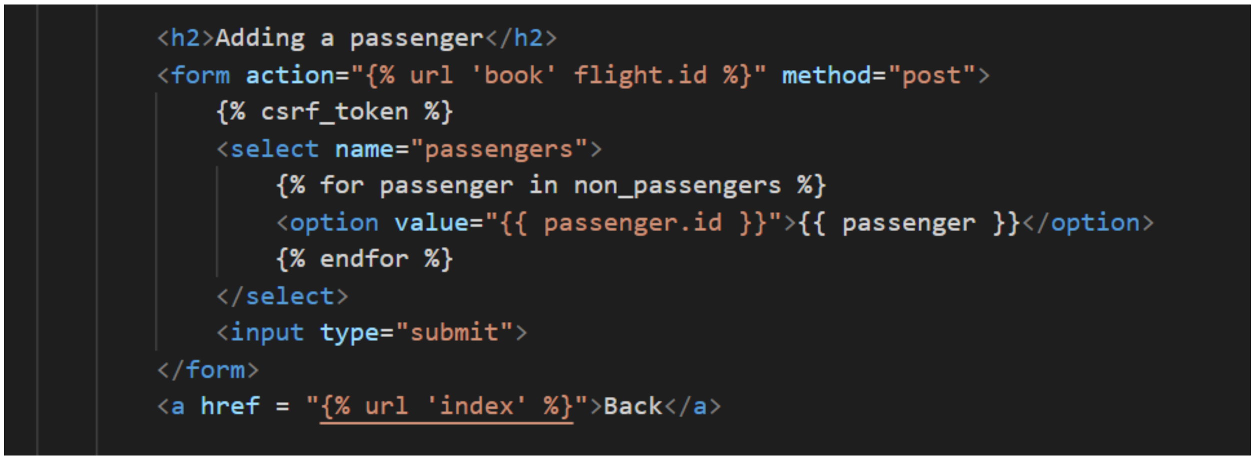
Flight=Flight.objects.get(pk=flight_id) You need the flight and passenger information

Request.POST[“Passenger”] it is the data about which passenger id we want to register on this flight is going to be passed in a form within input field name is passenger. إنها البيانات المتعلقة بأي راكب إذا أردنا التسجيل في هذه الرحلة التي سيتم تمريرها في نموذج داخل اسم حقل الإدخال هو الراكب.

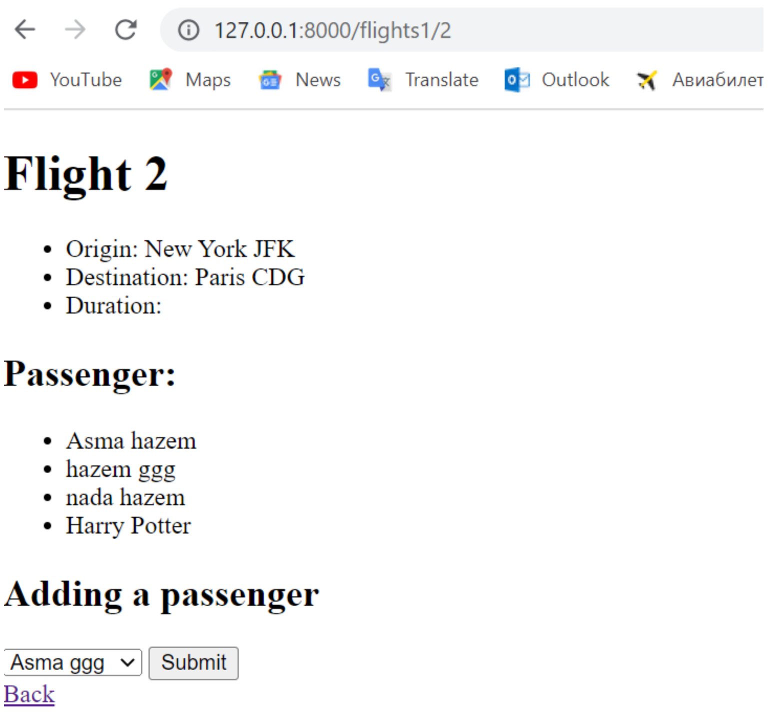
Passenger1.flights.add(flight) Inserting row to the passenger table

Return HttpResponseRedirect(reverse(“flight”,args=(flight_id))) redirect the page back to the flight.html page and with the argument which contains the flight id which was entered by the user أعد توجيه الصفحة مرة أخرى إلى صفحة flight.html وباستخدام الوسيطة التي تحتوي على معرف الرحلة الذي أدخله المستخدم

- Creating a form to book a flight for a passenger selected from a dropdown list:



```
<h2>Adding a passenger</h2>
<form action="{% url 'book' flight.id %}" method="post">
    {% csrf_token %}
    <select name="passengers">
        {% for passenger in non_passengers %}
        <option value="{{ passenger.id }}">{{ passenger }}</option>
        {% endfor %}
    </select>
    <input type="submit">
</form>
<a href = "{% url 'index' %}">Back</a>
```



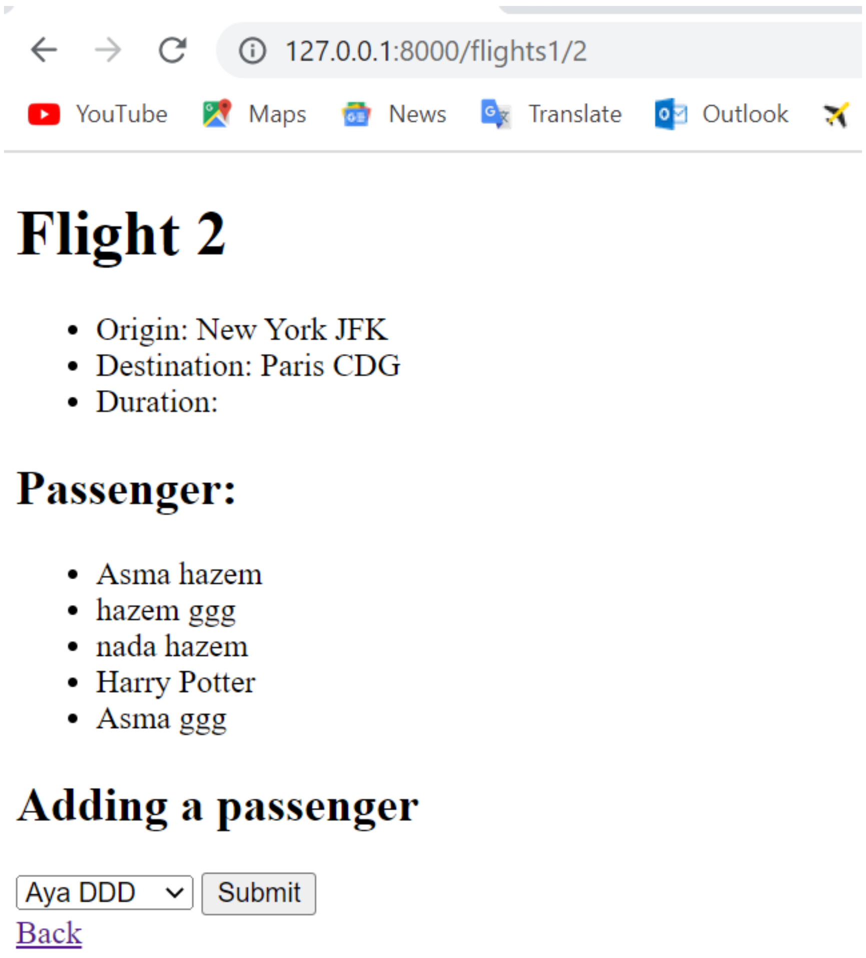
Flight 2

- Origin: New York JFK
- Destination: Paris CDG
- Duration:

Passenger:

- Asma hazem
- hazem ggg
- nada hazem
- Harry Potter

Adding a passenger


127.0.0.1:8000/flights1/2

Flight 2

- Origin: New York JFK
- Destination: Paris CDG
- Duration:

Passenger:

- Asma hazem
- hazem ggg
- nada hazem
- Harry Potter
- Asma ggg

Adding a passenger