API Implementation Explanation

Leshan Wang 2019110023

Introduction

The project follows Django restful architecture and simulates a simple composition of common APIs used by airline companies. The system is designed with APIs for Payment Service Providers, Airline Server, and client endpoints, under instructions and ideas of the former design document.

API Design

There are 17 APIs being implemented in total. Each provides distinct yet important feature to service users.

1. /users: This is a Django default API, used for debugging in during development phase.
2. /groups: This is a Django default API, used for debugging in during development phase.
3. /planes: This API is used by stuffs from airline companies to add detailed technical specifications of an airplane into the database. Plane model, number of seats, plane size, plane manufacturer, and operation age are required.
4. /airline: This API is used to search for airline company, it only takes the company name.
5. /findflight: This API is used to search for flights from on location to another, it requires a departure airport and an arrival airport.
6. /bookflight: This API is used to book a flight, it requires customer's name, the flight id, seat id, customer id, email address, and phone number.
7. /paymentmethod: This API is used to check available payment methods, it only needs a authorization code.
8. /payforbooking: This API is used to pay for the bookings, it requires the booking price, booking number, and booking method.

9. /finalize: This API is used to finalize the payment; it takes the booking number and an electronic stamp.

10. /status: This API is used to check the payment status; it only requires the booking number.

11. /cancel: This API is used to cancel bookings; it only requires the booking number.

12. /signin: This API is used for clients to login into the website, it requires the email address and password.

13. /sigunup: This API is used to let client signup, it takes the client's name, a username, email address, password, bank card number, bank account name, and phone number.

14. /deposit: This API is used to deposit money into the account, it needs an access token, the bank card number, and the password.

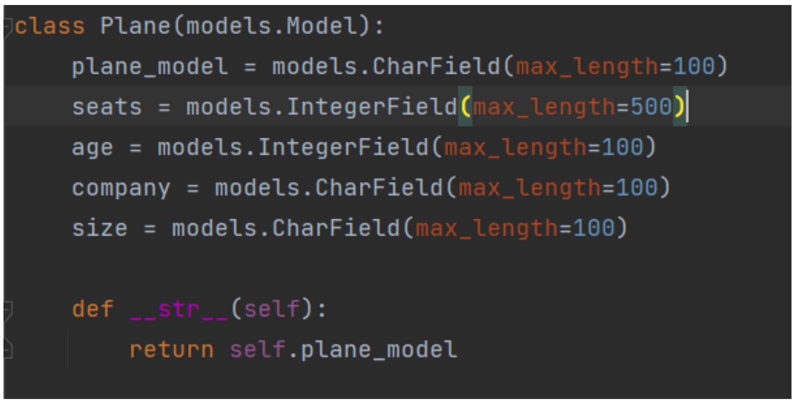
15. /pay: This API is used to make payment, it needs an access token, booking number, customer id, and ticket price.

16. /balance: This API is used to check balance; it only requires an access token.

17. /statement: This API is used to retrieve the statement; it only requires an access token.

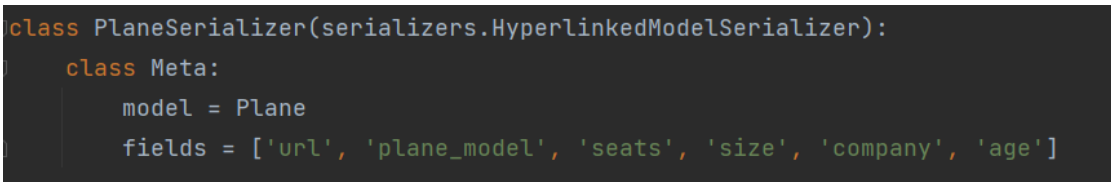
Implementation

The project is developed based on the default template of Django. Each API is composed by a base model, a serializer, a view set, and an URL. The required inputs are defined in the base model located in *models.py*. E.g. plane API.



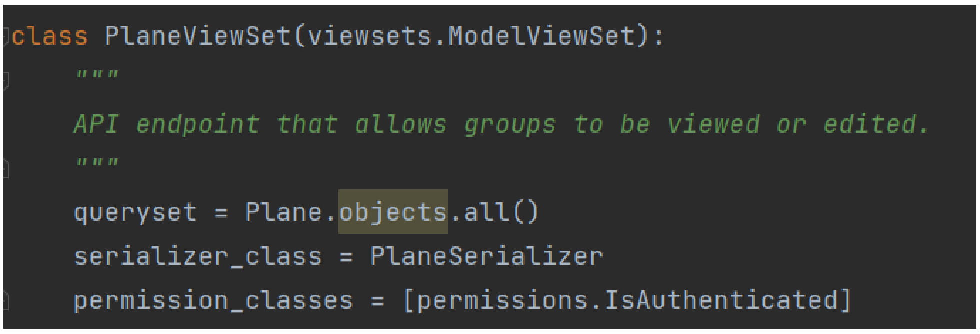
```python
class Plane(models.Model):
    plane_model = models.CharField(max_length=100)
    seats = models.IntegerField(max_length=500)
    age = models.IntegerField(max_length=100)
    company = models.CharField(max_length=100)
    size = models.CharField(max_length=100)

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

Then in *serializers.py*, the model is serialized using hyperlinked relations with *HyperLinkedModelSerializer*, which is a good restful design.



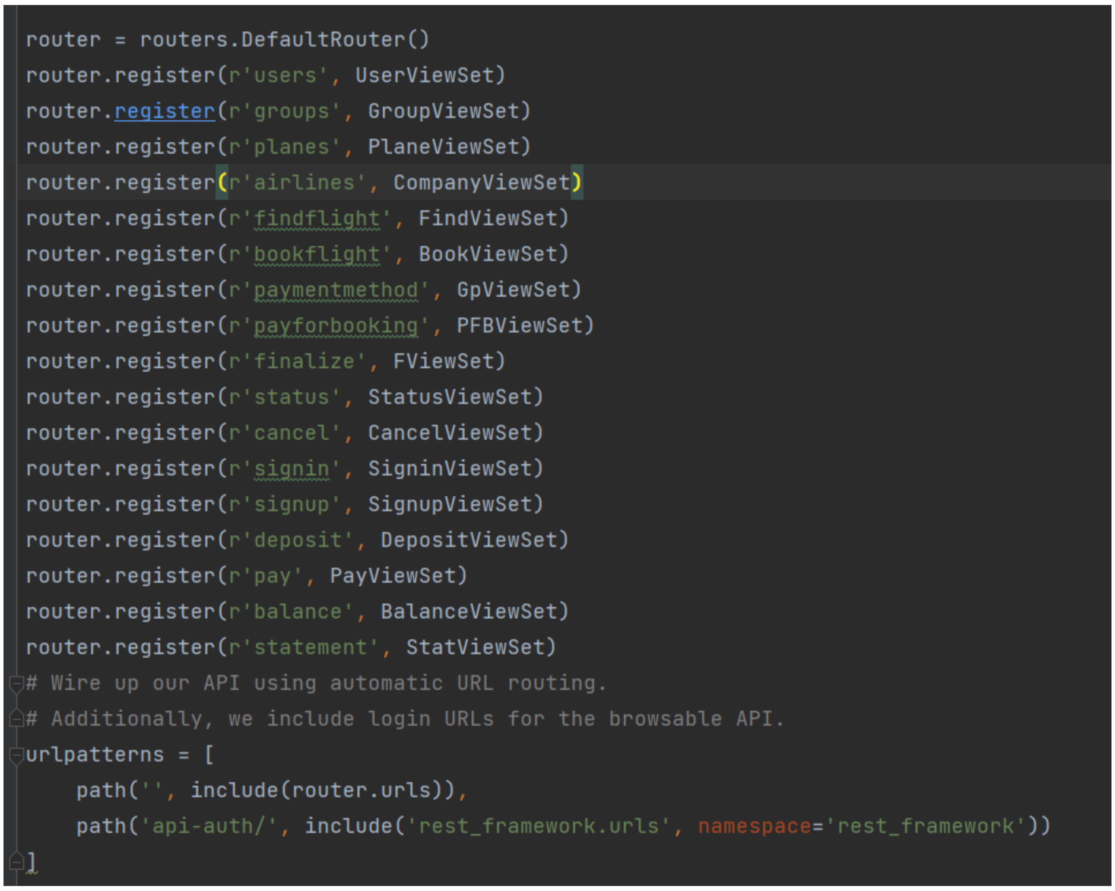
```python
class PlaneSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Plane
        fields = ['url', 'plane_model', 'seats', 'size', 'company', 'age']
```

Next, views are created in *views.py*, for each model, the common behaviors are grouped together into *ViewSets*; this keeps the view logic and nicely organized.



```python
class PlaneViewSet(viewsets.ModelViewSet):
    """
    API endpoint that allows groups to be viewed or edited.
    """
    queryset = Plane.objects.all()
    serializer_class = PlaneSerializer
    permission_classes = [permissions.IsAuthenticated]
```

Finally, The URL of each API is created inside urls.py, where *ViewSets* are used instead of views, the URL config can be automatically generated by registering *ViewSets* into router classes.

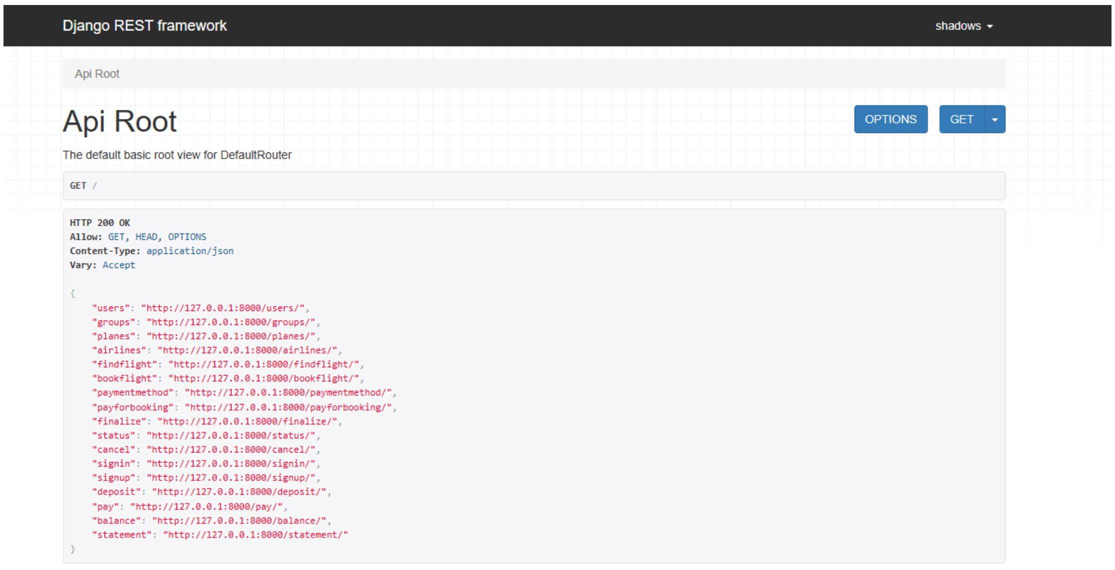


```python
router = routers.DefaultRouter()
router.register(r'users', UserViewSet)
router.register(r'groups', GroupViewSet)
router.register(r'planes', PlaneViewSet)
router.register(r'airlines', CompanyViewSet)
router.register(r'findflight', FindViewSet)
router.register(r'bookflight', BookViewSet)
router.register(r'paymentmethod', GpViewSet)
router.register(r'payforbooking', PFBViewSet)
router.register(r'finalize', FViewSet)
router.register(r'status', StatusViewSet)
router.register(r'cancel', CancelViewSet)
router.register(r'signin', SigninViewSet)
router.register(r'signup', SignupViewSet)
router.register(r'deposit', DepositViewSet)
router.register(r'pay', PayViewSet)
router.register(r'balance', BalanceViewSet)
router.register(r'statement', StatViewSet)
# Wire up our API using automatic URL routing.
# Additionally, we include login URLs for the browsable API.
urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls', namespace='rest_framework'))
]
```

The APIs are all created at this point.

Demonstration

Launch the program in local environment, user is required to login in order to interact with the APIs.

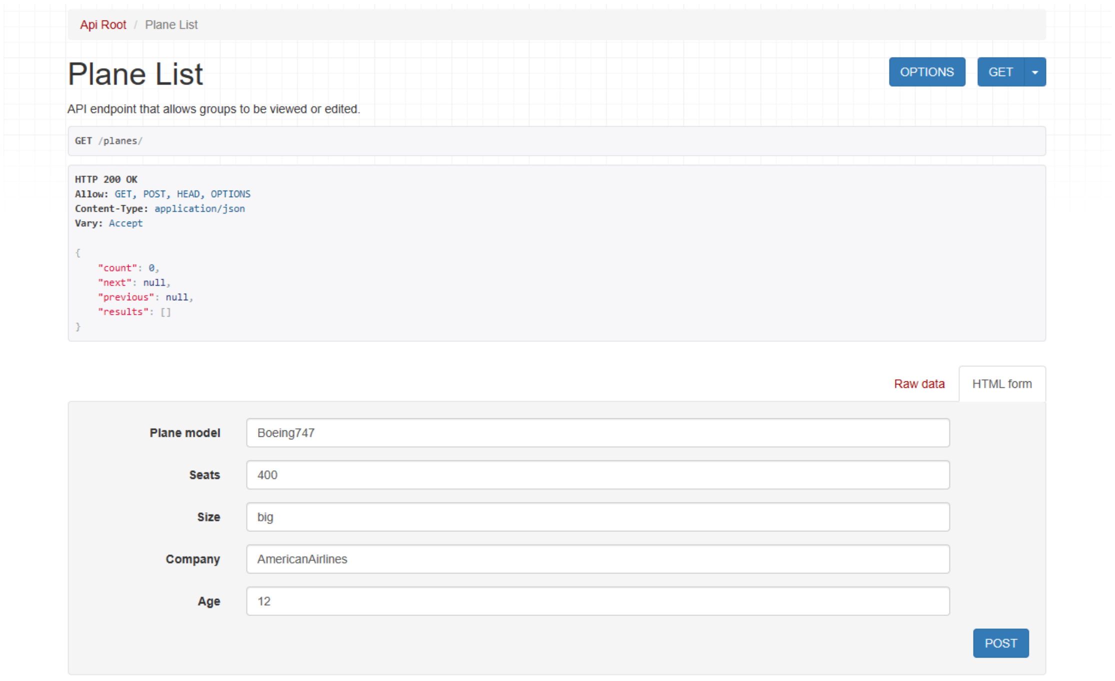


Api Root

The default basic root view for DefaultRouter

```
GET /
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "users": "http://127.0.0.1:8000/users/",
    "groups": "http://127.0.0.1:8000/groups/",
    "planes": "http://127.0.0.1:8000/planes/",
    "airlines": "http://127.0.0.1:8000/airlines/",
    "findflight": "http://127.0.0.1:8000/findflight/",
    "bookflight": "http://127.0.0.1:8000/bookflight/",
    "paymentmethod": "http://127.0.0.1:8000/paymentmethod/",
    "payforbooking": "http://127.0.0.1:8000/payforbooking/",
    "finalize": "http://127.0.0.1:8000/finalize/",
    "status": "http://127.0.0.1:8000/status/",
    "cancel": "http://127.0.0.1:8000/cancel/",
    "signin": "http://127.0.0.1:8000/signin/",
    "signup": "http://127.0.0.1:8000/signup/",
    "deposit": "http://127.0.0.1:8000/deposit/",
    "pay": "http://127.0.0.1:8000/pay/",
    "balance": "http://127.0.0.1:8000/balance/",
    "statement": "http://127.0.0.1:8000/statement/"
}
```



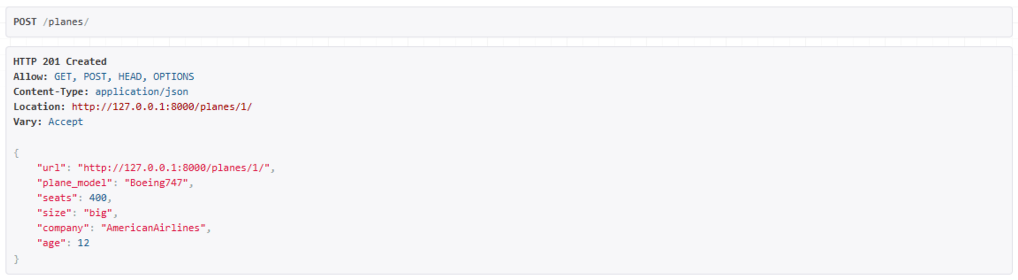
Plane List

API endpoint that allows groups to be viewed or edited.

```
GET /planes/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 0,
    "next": null,
    "previous": null,
    "results": []
}
```

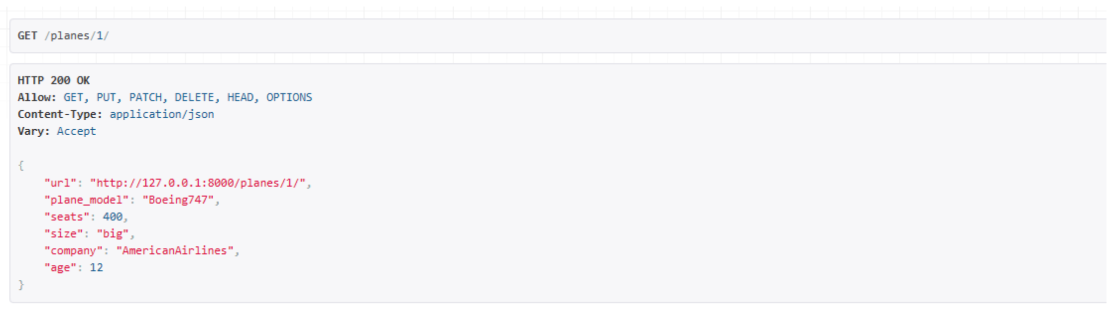
Plane model	Boeing747
Seats	400
Size	big
Company	AmericanAirlines
Age	12



```
POST /planes/
```

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Location: http://127.0.0.1:8000/planes/1/
Vary: Accept

{
    "url": "http://127.0.0.1:8000/planes/1/",
    "plane_model": "Boeing747",
    "seats": 400,
    "size": "big",
    "company": "AmericanAirlines",
    "age": 12
}
```

The data entered goes into the local database. Details can be viewed by inspecting the URL of each API.



```
GET /planes/1/
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "url": "http://127.0.0.1:8000/planes/1/",
    "plane_model": "Boeing747",
    "seats": 400,
    "size": "big",
    "company": "AmericanAirlines",
    "age": 12
}
```

Conclusion

The report explains the design and implementation of the airline service provider. The project is deployed at https://sc19l2w.pythonanywhere.com/, detailed instructions are included in the *readme.txt* file.