

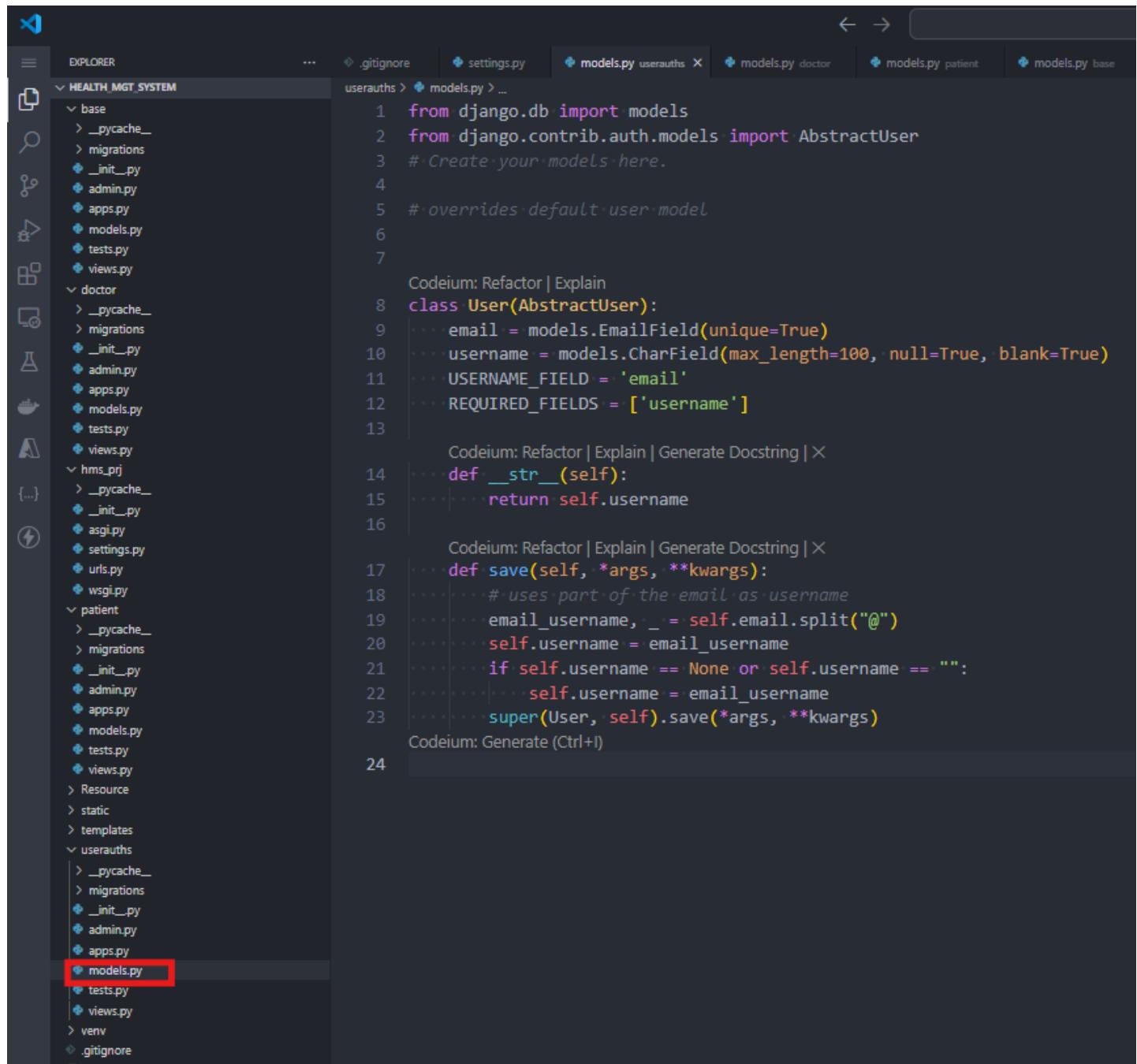
Topic: Models Part 2

Speaker: / Notebook: *Health Management System Using Django*



1. Unlike using the default USER MODEL, we use the ABSTRACTUSER class, so we can customize the default USER model. Update the userauths' app MODELS.PY:

COMMON FIELDS HERE USE 'blank = True' FOR THE PURPOSE OF IMPORTING THE DATA FROM CSV FILE, so as not to break our code.



```
1  from django.db import models
2  from django.contrib.auth.models import AbstractUser
3  # Create your models here.
4
5  # overrides default user model
6
7
8  class User(AbstractUser):
9      email = models.EmailField(unique=True)
10     username = models.CharField(max_length=100, null=True, blank=True)
11     USERNAME_FIELD = 'email'
12     REQUIRED_FIELDS = ['username']
13
14     def __str__(self):
15         return self.username
16
17     def save(self, *args, **kwargs):
18         # uses part of the email as username
19         email_username, _ = self.email.split("@")
20         self.username = email_username
21         if self.username == None or self.username == "":
22             self.username = email_username
23         super(User, self).save(*args, **kwargs)
24
Codeium: Generate (Ctrl+I)
```

2. Update the doctor's app MODELS.PY

```
from django.db import models
from userauths import models as userauths_models
from django.utils import timezone

NOTIFICATION_TYPE = (
    ("New Appointment", "New Appointment"),
    ("Appointment Canceled", "Appointment Canceled"),
)

class Doctor(models.Model):
    user = models.OneToOneField(
        userauths_models.User, on_delete=models.CASCADE)
    image = models.FileField(upload_to='images', blank=True, null=True)
    fullname = models.CharField(max_length=100, null=True, blank=True)
    mobile = models.CharField(max_length=100, null=True, blank=True)
    country = models.CharField(max_length=100, null=True, blank=True)
    bio = models.CharField(max_length=100, null=True, blank=True)
    specialization = models.CharField(max_length=100, null=True, blank=True)
    qualifications = models.CharField(max_length=100, null=True, blank=True)
    years_of_experience = models.CharField(
        max_length=100, null=True, blank=True)
    next_available_appointment_date = models.CharField(
        max_length=100, null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

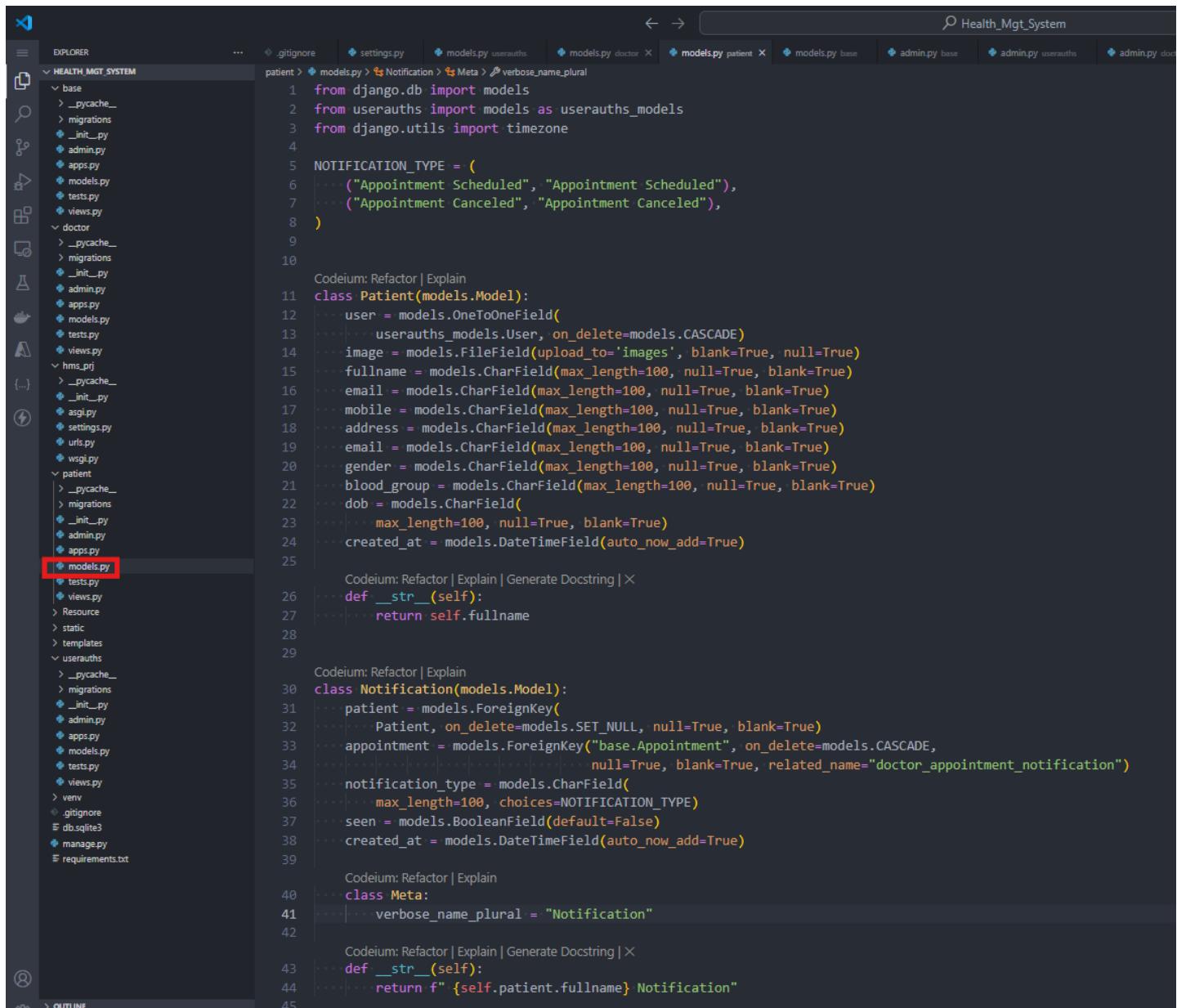
    def __str__(self):
        return f"Dr. {self.fullname}"

class Notification(models.Model):
    doctor = models.ForeignKey(
        Doctor, on_delete=models.SET_NULL, null=True, blank=True)
    appointment = models.ForeignKey("base.Appointment", on_delete=models.CASCADE,
                                   null=True, blank=True, related_name="doctor_appointment_notification")
    notification_type = models.CharField(
        max_length=100, choices=NOTIFICATION_TYPE)
    seen = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name_plural = "Notification"

    def __str__(self):
        return f"Dr. {self.doctor.fullname} Notification"
```

3. Update the patient's app MODELS.PY



```
patient > models.py > Notification > Meta > verbose_name_plural
1  from django.db import models
2  from userauths import models as userauths_models
3  from django.utils import timezone
4
5  NOTIFICATION_TYPE = (
6      ("Appointment Scheduled", "Appointment Scheduled"),
7      ("Appointment Canceled", "Appointment Canceled"),
8  )
9
10
11 class Patient(models.Model):
12     user = models.OneToOneField(
13         userauths_models.User, on_delete=models.CASCADE)
14     image = models.FileField(upload_to='images', blank=True, null=True)
15     fullname = models.CharField(max_length=100, null=True, blank=True)
16     email = models.CharField(max_length=100, null=True, blank=True)
17     mobile = models.CharField(max_length=100, null=True, blank=True)
18     address = models.CharField(max_length=100, null=True, blank=True)
19     gender = models.CharField(max_length=100, null=True, blank=True)
20     blood_group = models.CharField(max_length=100, null=True, blank=True)
21     dob = models.CharField(
22         max_length=100, null=True, blank=True)
23     created_at = models.DateTimeField(auto_now_add=True)
24
25
26 def __str__(self):
27     return self.fullname
28
29
30 class Notification(models.Model):
31     patient = models.ForeignKey(
32         Patient, on_delete=models.SET_NULL, null=True, blank=True)
33     appointment = models.ForeignKey("base.Appointment", on_delete=models.CASCADE,
34         null=True, blank=True, related_name="doctor_appointment_notification")
35     notification_type = models.CharField(
36         max_length=100, choices=NOTIFICATION_TYPE)
37     seen = models.BooleanField(default=False)
38     created_at = models.DateTimeField(auto_now_add=True)
39
40 class Meta:
41     verbose_name_plural = "Notification"
42
43 def __str__(self):
44     return f" {self.patient.fullname} Notification"
45
```

4. Update the base's app MODELS.PY

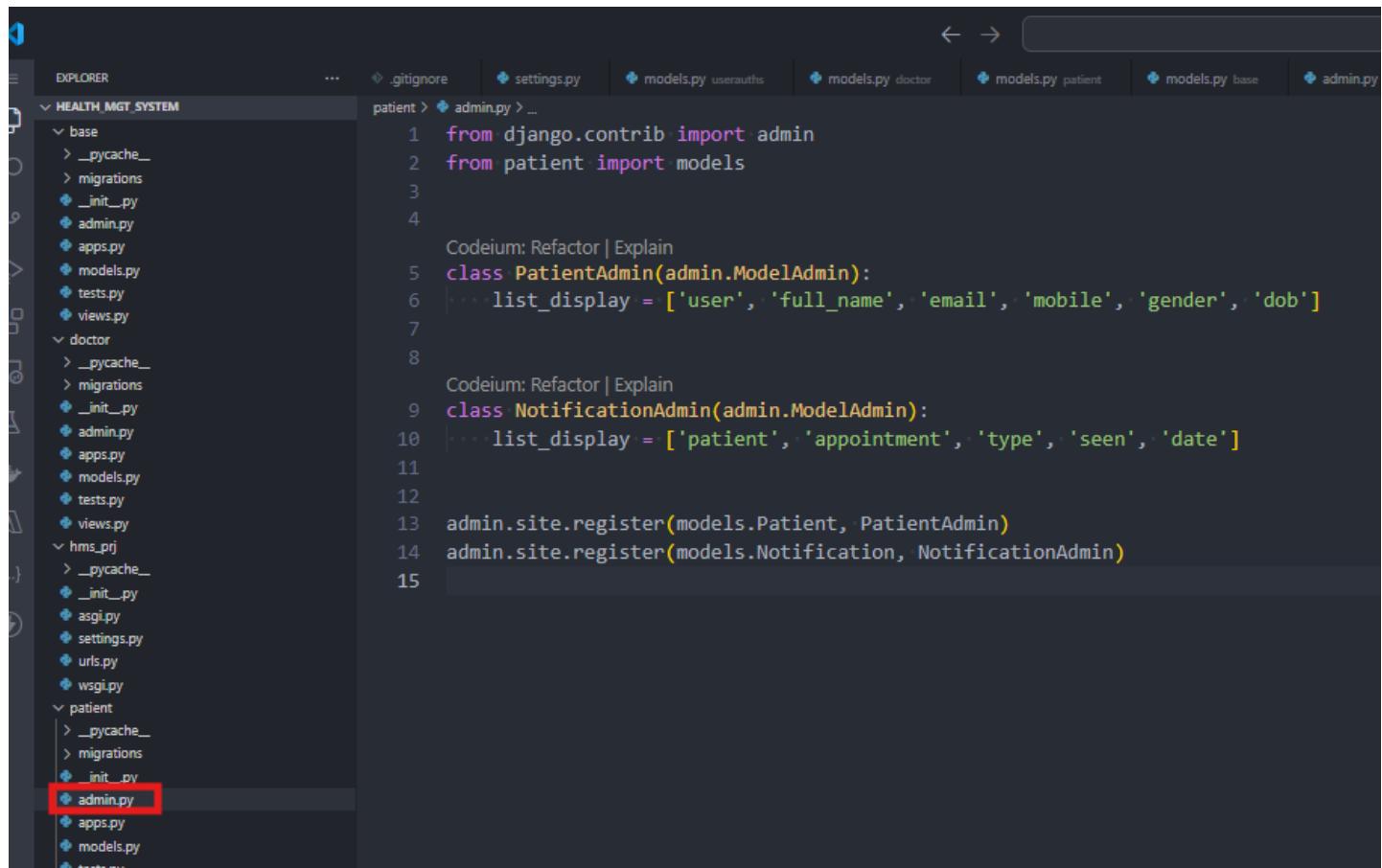
The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure of 'HEALTH_MGT_SYSTEM'. The 'models.py' file under the 'base' directory is the active tab, with its content displayed in the main editor area. The code defines two models: 'Service' and 'Appointment'. The 'Service' model has fields for image, name, description, cost, and a many-to-many relationship with 'Doctor'. The 'Appointment' model has fields for service, doctor, patient, appointment_date, issues, symptoms, appointment_id, and status. Code completion suggestions are visible in the code editor, such as 'Codeium: Refactor | Explain' and 'Codeium: Refactor | Explain | Generate Docstring'.

```
1  from django.db import models
2  from shortuuid.django_fields import ShortUUIDField
3
4  from doctor import models as doctor_models
5  from patient import models as patient_models
6
7
8  class Service(models.Model):
9      image = models.ImageField(upload_to='images', blank=True, null=True)
10     name = models.CharField(max_length=100)
11     description = models.TextField(null=True, blank=True)
12     cost = models.DecimalField(max_digits=10, decimal_places=2)
13     available_doctors = models.ManyToManyField(
14         doctor_models.Doctor, blank=True)
15
16     Codeium: Refactor | Explain | Generate Docstring | X
17     def __str__(self):
18         return f'{self.name} - {self.cost}'
19
20
21     class Appointment(models.Model):
22         STATUS = [
23             ('Scheduled', 'Scheduled'),
24             ('Completed', 'Completed'),
25             ('Pending', 'Pending'),
26             ('Cancelled', 'Cancelled')
27         ]
28
29         service = models.ForeignKey(Service, on_delete=models.SET_NULL,
30             null=True, blank=True, related_name='service_appointments')
31         doctor = models.ForeignKey(doctor_models.Doctor, on_delete=models.SET_NULL,
32             null=True, blank=True, related_name='doctor_appointments')
33         patient = models.ForeignKey(patient_models.Patient, on_delete=models.SET_NULL,
34             null=True, blank=True, related_name='appointments_patient')
35         appointment_date = models.DateTimeField(null=True, blank=True)
36         issues = models.TextField(blank=True, null=True)
37         symptoms = models.TextField(blank=True, null=True)
38         appointment_id = ShortUUIDField(
39             length=6, max_length=10, alphabet="1234567890")
40         status = models.CharField(max_length=120, choices=STATUS)
41
42         Codeium: Refactor | Explain | Generate Docstring | X
43         def __str__(self):
44             return f'{self.patient.full_name} with {self.doctor.full_name}'
```

```
45 class MedicalRecord(models.Model):
46     appointment = models.ForeignKey(Appointment, on_delete=models.CASCADE)
47     diagnosis = models.TextField()
48     treatment = models.TextField()
49
50     Codeium: Refactor | Explain | Generate Docstring | X
51     def __str__(self):
52         return f"Medical Record for {self.appointment.patient.full_name}"
53
54 class LabTest(models.Model):
55     appointment = models.ForeignKey(Appointment, on_delete=models.CASCADE)
56     test_name = models.CharField(max_length=255)
57     description = models.TextField(blank=True, null=True)
58     result = models.TextField(blank=True, null=True)
59
60     Codeium: Refactor | Explain | Generate Docstring | X
61     def __str__(self):
62         return f"{self.test_name}"
63
64 class Prescription(models.Model):
65     appointment = models.ForeignKey(Appointment, on_delete=models.CASCADE)
66     medications = models.TextField(blank=True, null=True)
67
68     Codeium: Refactor | Explain | Generate Docstring | X
69     def __str__(self):
70         return f"Prescription for {self.appointment.patient.full_name}"
71
72 class Billing(models.Model):
73     patient = models.ForeignKey(patient_models.Patient, on_delete=models.SET_NULL,
74                                 null=True, blank=True, related_name='billings')
75     appointment = models.ForeignKey(
76         Appointment, on_delete=models.CASCADE, related_name='billing', blank=True, null=True)
77     sub_total = models.DecimalField(max_digits=10, decimal_places=2)
78     tax = models.DecimalField(max_digits=10, decimal_places=2)
79     total = models.DecimalField(max_digits=10, decimal_places=2)
80     status = models.CharField(max_length=120, choices=[
81         ('Paid', 'Paid'), ('Unpaid', 'Unpaid')])
82     billing_id = ShortUUIDField(length=6, max_length=10, alphabet="1234567890")
83
84     date = models.DateTimeField(auto_now_add=True)
85
86     Codeium: Refactor | Explain | Generate Docstring | X
87     def __str__(self):
88         return f"Billing for {self.patient.full_name} - Total: {self.total}"
```

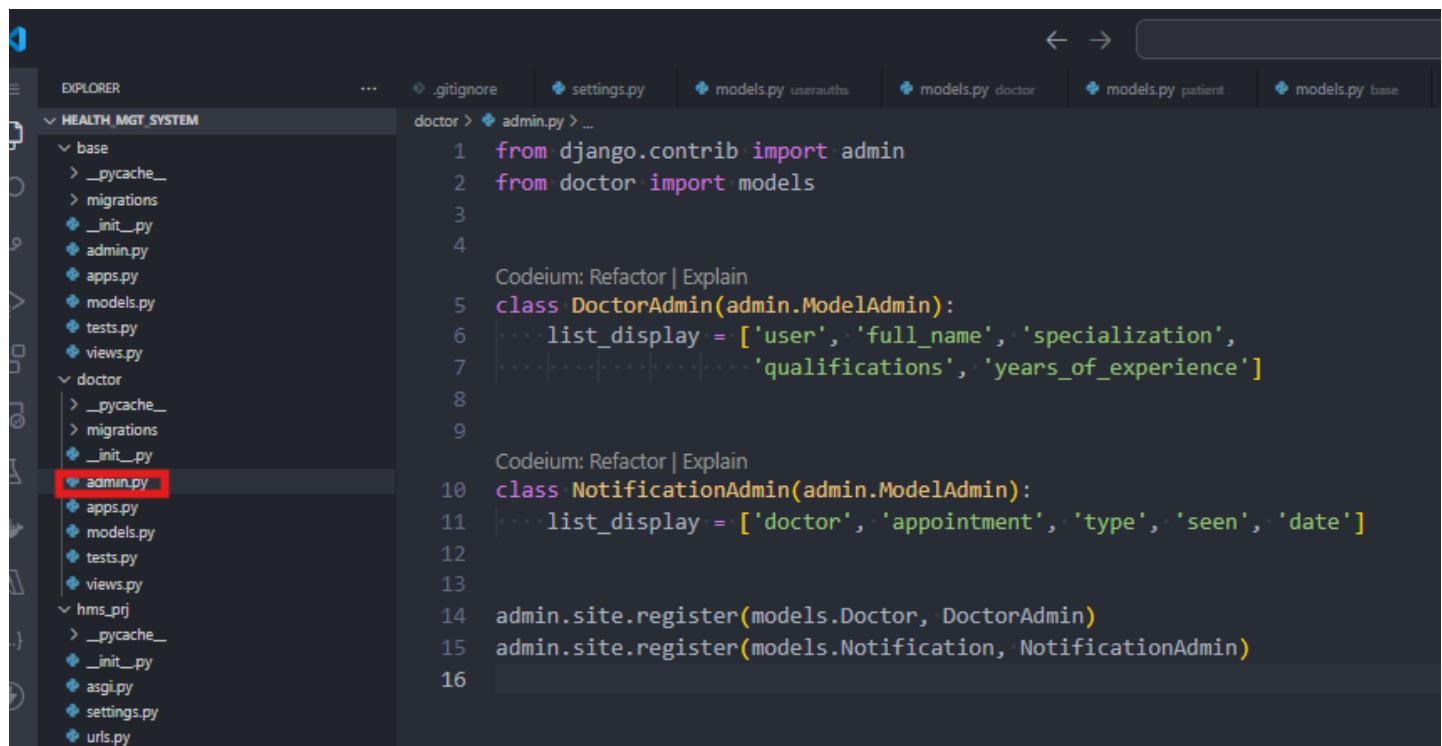
5. Update the ADMIN.PY

PATIENTS - ADMIN.PY



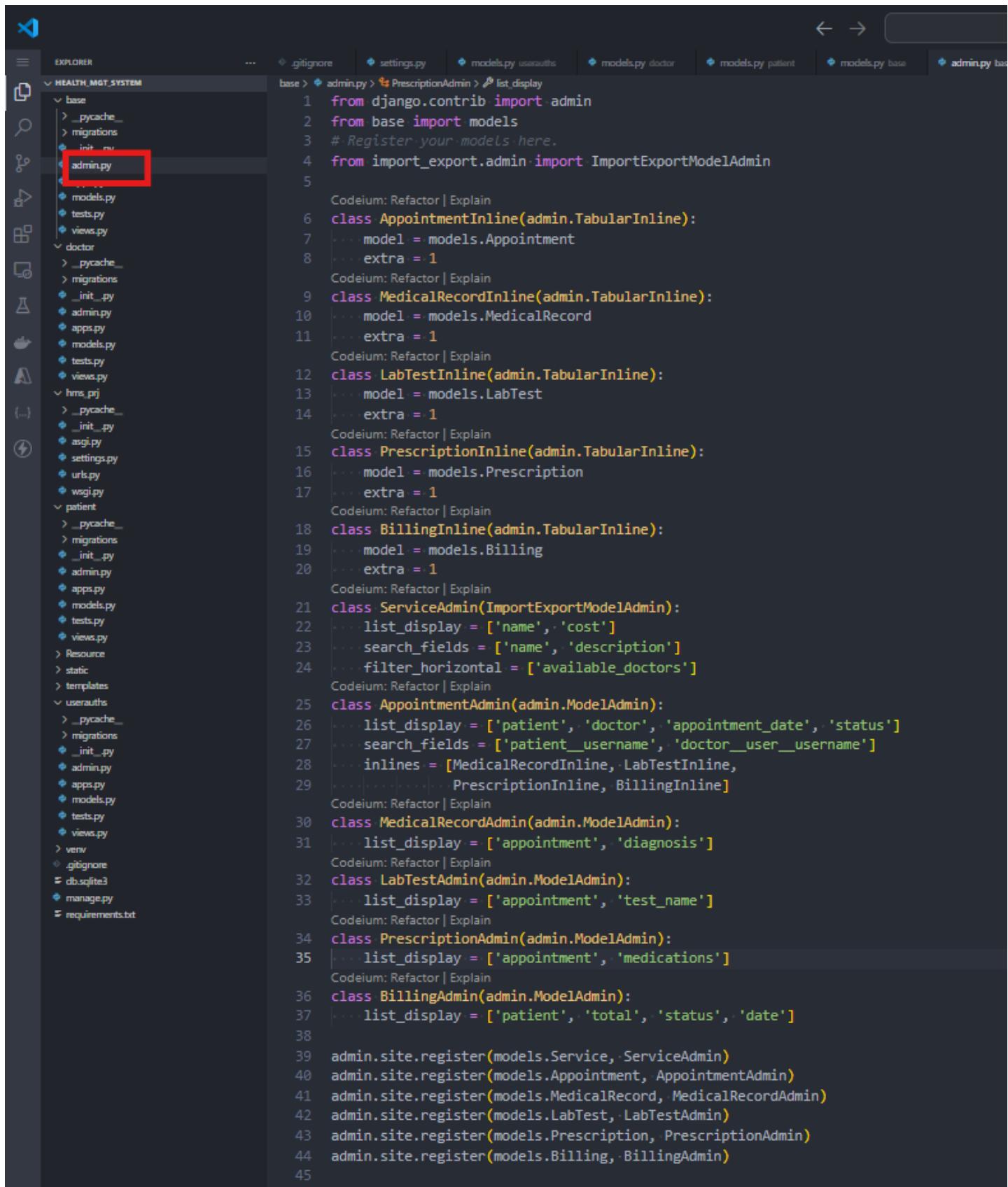
```
patient > admin.py > ...
1  from django.contrib import admin
2  from patient import models
3
4
5  class PatientAdmin(admin.ModelAdmin):
6      list_display = ['user', 'full_name', 'email', 'mobile', 'gender', 'dob']
7
8
9  class NotificationAdmin(admin.ModelAdmin):
10    list_display = ['patient', 'appointment', 'type', 'seen', 'date']
11
12
13 admin.site.register(models.Patient, PatientAdmin)
14 admin.site.register(models.Notification, NotificationAdmin)
15
```

DOCTORS - ADMIN.PY



```
doctor > admin.py > ...
1  from django.contrib import admin
2  from doctor import models
3
4
5  class DoctorAdmin(admin.ModelAdmin):
6      list_display = ['user', 'full_name', 'specialization',
7                      'qualifications', 'years_of_experience']
8
9
10 class NotificationAdmin(admin.ModelAdmin):
11    list_display = ['doctor', 'appointment', 'type', 'seen', 'date']
12
13
14 admin.site.register(models.Doctor, DoctorAdmin)
15 admin.site.register(models.Notification, NotificationAdmin)
16
```

BASE - ADMIN.PY



```

base > admin.py > PrescriptionAdmin > list_display
1  from django.contrib import admin
2  from base import models
3  # Register your models here.
4  from import_export.admin import ImportExportModelAdmin
5
6  class AppointmentInline(admin.TabularInline):
7      model = models.Appointment
8      extra = 1
9  class MedicalRecordInline(admin.TabularInline):
10     model = models.MedicalRecord
11     extra = 1
12  class LabTestInline(admin.TabularInline):
13     model = models.LabTest
14     extra = 1
15  class PrescriptionInline(admin.TabularInline):
16     model = models.Prescription
17     extra = 1
18  class BillingInline(admin.TabularInline):
19     model = models.Billing
20     extra = 1
21  class ServiceAdmin(ImportExportModelAdmin):
22      list_display = ['name', 'cost']
23      search_fields = ['name', 'description']
24      filter_horizontal = ['available_doctors']
25  class AppointmentAdmin(admin.ModelAdmin):
26      list_display = ['patient', 'doctor', 'appointment_date', 'status']
27      search_fields = ['patient__username', 'doctor__user__username']
28      inlines = [MedicalRecordInline, LabTestInline,
29                 PrescriptionInline, BillingInline]
29
30  class MedicalRecordAdmin(admin.ModelAdmin):
31      list_display = ['appointment', 'diagnosis']
32  class LabTestAdmin(admin.ModelAdmin):
33      list_display = ['appointment', 'test_name']
34  class PrescriptionAdmin(admin.ModelAdmin):
35      list_display = ['appointment', 'medications']
36  class BillingAdmin(admin.ModelAdmin):
37      list_display = ['patient', 'total', 'status', 'date']
38
39  admin.site.register(models.Service, ServiceAdmin)
40  admin.site.register(models.Appointment, AppointmentAdmin)
41  admin.site.register(models.MedicalRecord, MedicalRecordAdmin)
42  admin.site.register(models.LabTest, LabTestAdmin)
43  admin.site.register(models.Prescription, PrescriptionAdmin)
44  admin.site.register(models.Billing, BillingAdmin)
45

```

6. We commented out PATIENTS AND DOCTORS and BASE MODELS.PY AND ADMIN.PY and ran migrations app by app, so as not to have dependency conflict (CIRCULAR DEPENDENCY i.e. doctor is dependent to base, base is dependent to doctor.)

7. We first did our makemigrations and migrate command on USERAUTHS. It was causing us an error, so we removed the SQLITE and everything in the MIGRATIONS folder of USERAUTHS EXCEPT THE __init__.py

1. Reset the SQLite database (Optional)

If your project is new and you don't have any important data in your database, you can delete the `db.sqlite3` file and the migrations files in each of your apps (in the `migrations` folder, excluding the `__init__.py` file), and then rerun the migrations. Here's how:

- Delete `db.sqlite3`:

```
bash
```

 Copy code

```
rm db.sqlite3
```

- Delete migration files: Go into each app's `migrations/` folder and delete all migration files except `__init__.py`.

Steps to Clean Up Migrations

1. Delete Migration Files:

- Go into each app's `migrations/` folder.
- Delete all migration files except for `__init__.py`.

2. Delete `__pycache__` Folder:

- Inside each app's `migrations/` folder, delete the `__pycache__` folder if it exists.

3. Delete the Database:

- If you're starting fresh and don't need any data, delete `db.sqlite3`.

4. Rerun Migrations:

- After cleaning up, run:

```
bash
```

 Copy code

```
python manage.py makemigrations
python manage.py migrate
```

We run our migrations and USERAUTHS model works now. We then add our migrations on BASE app.

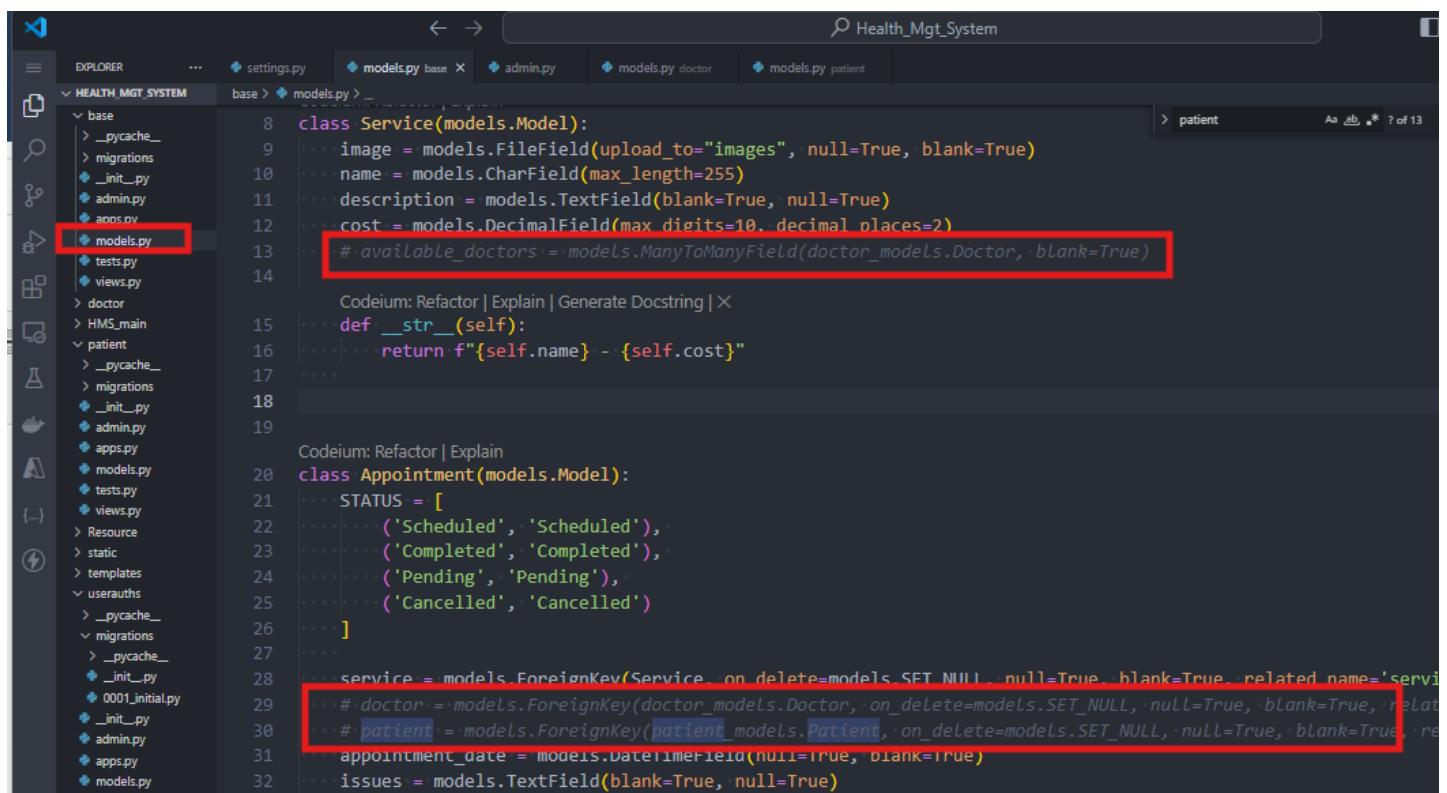
The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- Project Explorer:** Shows the project structure under `HEALTH_MGT_SYSTEM`. A red box highlights the `userauths` directory and its sub-directories (`__pycache__`, `migrations`, `__init__.py`, `0001_initial.py`).
- Code Editor:** The `admin.py` file is open, showing code related to the `base` app. It includes imports for `models` and `ImportExportModelAdmin`, and definitions for `AppointmentInline`, `MedicalRecordInline`, and `LabTestInline` (each extending `TabularInline`).
- Terminal:** The terminal window shows the following commands and output:
 - `$ rm db.sqlite3` (venv)
 - `$ python manage.py makemigrations userauths`
Migrations for 'userauths':
userauths/migrations/0001_initial.py
- Create model User
 - `$ python manage.py migrate userauths`
Operations to perform:
Apply all migrations: userauths
Running migrations:
Applying contenttypes.0001_initial... OK
Applying contenttypes.0002_remove_content_type_name... OK
Applying auth.0001_initial... OK
Applying auth.0002.Alter_permission_name_max_length... OK
Applying auth.0003.Alter_user_email_max_length... OK
Applying auth.0004.Alter_user_username_opts... OK
Applying auth.0005.Alter_user_last_login_null... OK
Applying auth.0006.Require_contenttypes_0002... OK
Applying auth.0007.Alter_validators_add_error_messages... OK
Applying auth.0008.Alter_user_username_max_length... OK
Applying auth.0009.Alter_user_last_name_max_length... OK
Applying auth.0010.Alter_group_name_max_length... OK
Applying auth.0011.Update_proxy_permissions... OK
Applying auth.0012.Alter_user_first_name_max_length... OK
Applying userauths.0001_initial... OK

Since we commented out DOCTORS AND PATIENT model.py and admin.py, We undo this and execute our migrations next on them.

```
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/M
● $ python manage.py migrate doctor
Operations to perform:
  Apply all migrations: doctor
Running migrations:
  Applying doctor.0001_initial... OK
(venv)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/M
● $ python manage.py makemigrations patient
Migrations for 'patient':
  patient\migrations\0001_initial.py
    - Create model Patient
    - Create model Notification
(venv)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/M
● $ python manage.py migrate patient
Operations to perform:
  Apply all migrations: patient
Running migrations:
  Applying patient.0001_initial... OK
(venv)
```

We then remove the commented lines in our BASE APP\MODELS.PY



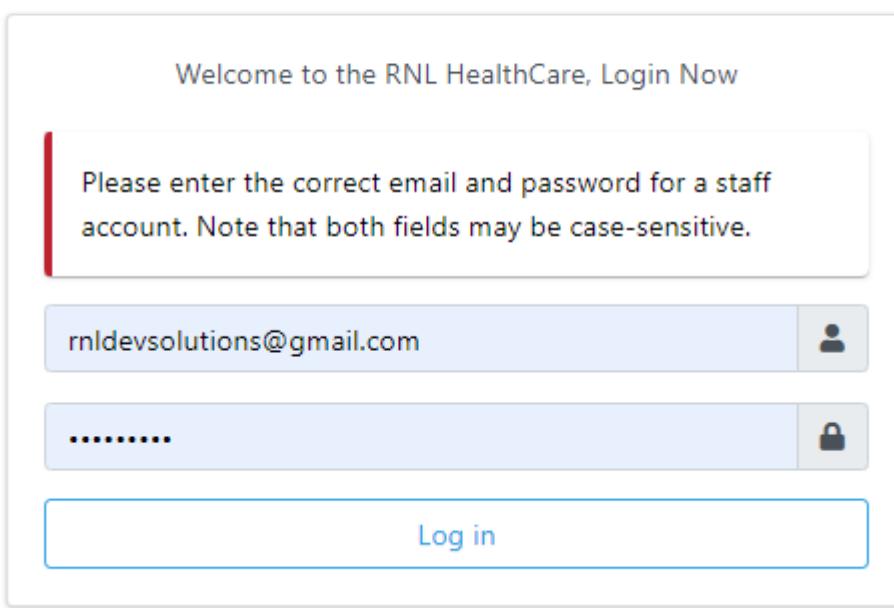
```
8  class Service(models.Model):
9      image = models.FileField(upload_to="images", null=True, blank=True)
10     name = models.CharField(max_length=255)
11     description = models.TextField(blank=True, null=True)
12     cost = models.DecimalField(max_digits=10, decimal_places=2)
13     # available_doctors = models.ManyToManyField(doctor_models.Doctor, blank=True)
14
15     def __str__(self):
16         return f"{self.name} - {self.cost}"
17
18
19
20 class Appointment(models.Model):
21     STATUS = [
22         ('Scheduled', 'Scheduled'),
23         ('Completed', 'Completed'),
24         ('Pending', 'Pending'),
25         ('Cancelled', 'Cancelled')
26     ]
27
28     service = models.ForeignKey(Service, on_delete=models.SET_NULL, null=True, blank=True, related_name='services')
29     # doctor = models.ForeignKey(doctor_models.Doctor, on_delete=models.SET_NULL, null=True, blank=True, related_name='patients')
30     # patient = models.ForeignKey(patient_models.Patient, on_delete=models.SET_NULL, null=True, blank=True, related_name='appointments')
31     appointment_date = models.DateTimeField(null=True, blank=True)
32     issues = models.TextField(blank=True, null=True)
```

We then run the general migrations for all the default models of Django project.

```
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
$ python manage.py makemigrations base
Migrations for 'base':
  base\migrations\0002_appointment_doctor_appointment_patient_and_more.py
    - Add field doctor to appointment
    - Add field patient to appointment
    - Add field patient to billing
    - Add field available_doctors to service
(venv)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, base, contenttypes, doctor, patient, sessions, userauths
Running migrations:
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying base.0002_appointment_doctor_appointment_patient_and_more... OK
  Applying sessions.0001_initial... OK
(venv)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
$ python manage.py makemigrations
• No changes detected
(venv)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
$ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, base, contenttypes, doctor, patient, sessions, userauths
Running migrations:
  No migrations to apply
```

Create a new superuser now.

Run the server again and use the email as the login credentials. This Login interface shows up instead.



When you remove the comments around ADMIN.PY of BASE, PATIENT, DOCTOR, the ADMIN DASHBOARD now looks like this:

http://127.0.0.1:8000/admin/

Recent actions
None available

When you open the model like SERVICE, its interface shall be:

Services | Home > Base > Services > Add service

Image Choose File No file chosen

Name *

Description

Cost *

Available doctors

Available available doctors		Chosen available doctors	
Filter		Filter	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Save
Save and add another
Save and continue editing