

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
```

2. Update the doctor's app MODELS.PY

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'HEALTH_MGT_SYSTEM' with a 'doctor' app. The code editor shows the following Python code:

```

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     ("New Appointment", "New Appointment"),
7     ("Appointment Canceled", "Appointment Canceled"),
8 )
9
10
11 Codeium: Refactor | Explain
12 class Doctor(models.Model):
13     user = models.OneToOneField(
14         userauths_models.User, on_delete=models.CASCADE)
15     image = models.FileField(upload_to='images', blank=True, null=True)
16     fullname = models.CharField(max_length=100, null=True, blank=True)
17     mobile = models.CharField(max_length=100, null=True, blank=True)
18     country = models.CharField(max_length=100, null=True, blank=True)
19     bio = models.CharField(max_length=100, null=True, blank=True)
20     specialization = models.CharField(max_length=100, null=True, blank=True)
21     qualifications = models.CharField(max_length=100, null=True, blank=True)
22     years_of_experience = models.CharField(
23         max_length=100, null=True, blank=True)
24     next_available_appointment_date = models.CharField(
25         max_length=100, null=True, blank=True)
26     created_at = models.DateTimeField(auto_now_add=True)
27
28     Codeium: Refactor | Explain | Generate Docstring | X
29     def __str__(self):
30         return f"Dr. {self.fullname}"
31
32 Codeium: Refactor | Explain
33 class Notification(models.Model):
34     doctor = models.ForeignKey(
35         Doctor, on_delete=models.SET_NULL, null=True, blank=True)
36     appointment = models.ForeignKey("base.Appointment", on_delete=models.CASCADE,
37         null=True, blank=True, related_name="doctor_appointment_notification")
38     notification_type = models.CharField(
39         max_length=100, choices=NOTIFICATION_TYPE)
40     seen = models.BooleanField(default=False)
41     created_at = models.DateTimeField(auto_now_add=True)
42
43     Codeium: Refactor | Explain
44     class Meta:
45         verbose_name_plural = "Notification"
46
47     Codeium: Refactor | Explain | Generate Docstring | X
48     def __str__(self):
49         return f"Dr. {self.doctor.fullname} Notification"

```

3. Update the patient's app MODELS.PY

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'HEALTH_MGT_SYSTEM' with several subdirectories: 'base', 'doctor', 'patient', 'userauths', and 'venv'. The 'patient' directory is selected, and the 'models.py' file is highlighted. The code editor shows the following Python code:

```
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 Codeium: Refactor | Explain
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     email = models.CharField(max_length=100, null=True, blank=True)
20     gender = models.CharField(max_length=100, null=True, blank=True)
21     blood_group = models.CharField(max_length=100, null=True, blank=True)
22     dob = models.CharField(
23         max_length=100, null=True, blank=True)
24     created_at = models.DateTimeField(auto_now_add=True)
25
26     Codeium: Refactor | Explain | Generate Docstring | X
27     def __str__(self):
28         return self.fullname
29
30 Codeium: Refactor | Explain
31 class Notification(models.Model):
32     patient = models.ForeignKey(
33         Patient, on_delete=models.SET_NULL, null=True, blank=True)
34     appointment = models.ForeignKey("base.Appointment", on_delete=models.CASCADE,
35         null=True, blank=True, related_name="doctor_appointment_notification")
36     notification_type = models.CharField(
37         max_length=100, choices=NOTIFICATION_TYPE)
38     seen = models.BooleanField(default=False)
39     created_at = models.DateTimeField(auto_now_add=True)
40
41     Codeium: Refactor | Explain
42     class Meta:
43         verbose_name_plural = "Notification"
44
45     Codeium: Refactor | Explain | Generate Docstring | X
46     def __str__(self):
47         return f" {self.patient.fullname} Notification"
```

4. Update the base's app MODELS.PY

```
EXPLORER  ...  gitignore  settings.py  models.py userauths  models.py doctor  models.py patient  models.py base x  admin.py base  admin.py us

HEALTH_MGT_SYSTEM  base > models.py > ...
  base
  > __pycache__
  > migrations
  > __init__.py
  > admin.py
  > apps.py
  > models.py
  > tests.py
  > views.py
  doctor
  > __pycache__
  > migrations
  > __init__.py
  > admin.py
  > apps.py
  > models.py
  > tests.py
  > views.py
  hms_prij
  > __pycache__
  > __init__.py
  > asgi.py
  > settings.py
  > urls.py
  > wsgi.py
  patient
  > __pycache__
  > migrations
  > __init__.py
  > admin.py
  > apps.py
  > models.py
  > tests.py
  > views.py
  > Resource
  > static
  > templates
  > userauths
  > __pycache__
  > migrations
  > __init__.py
  > admin.py
  > apps.py
  > models.py
  > tests.py
  > views.py
  > venv
  > .gitignore
  > db.sqlite3
  > manage.py
  > requirements.txt

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  Codeium: Refactor | Explain
9  class Service(models.Model):
10     image = models.FileField(upload_to='images', blank=True, null=True)
11     name = models.CharField(max_length=100)
12     description = models.TextField(null=True, blank=True)
13     cost = models.DecimalField(max_digits=10, decimal_places=2)
14     available_doctors = models.ManyToManyField(
15         doctor_models.Doctor, blank=True)
16
17     Codeium: Refactor | Explain | Generate Docstring | X
18     def __str__(self):
19         return f"{self.name} - {self.cost}"
20
21  Codeium: Refactor | Explain
22  class Appointment(models.Model):
23     STATUS = [
24         ('Scheduled', 'Scheduled'),
25         ('Completed', 'Completed'),
26         ('Pending', 'Pending'),
27         ('Cancelled', 'Cancelled')
28     ]
29
30     service = models.ForeignKey(Service, on_delete=models.SET_NULL,
31         null=True, blank=True, related_name='service_appointments')
32     doctor = models.ForeignKey(doctor_models.Doctor, on_delete=models.SET_NULL,
33         null=True, blank=True, related_name='doctor_appointments')
34     patient = models.ForeignKey(patient_models.Patient, on_delete=models.SET_NULL,
35         null=True, blank=True, related_name='appointments_patient')
36     appointment_date = models.DateTimeField(null=True, blank=True)
37     issues = models.TextField(blank=True, null=True)
38     symptoms = models.TextField(blank=True, null=True)
39     appointment_id = ShortUUIDField(
40         length=6, max_length=10, alphabet="1234567890")
41     status = models.CharField(max_length=120, choices=STATUS)
42
43     Codeium: Refactor | Explain | Generate Docstring | X
44     def __str__(self):
45         return f"{self.patient.full_name} with {self.doctor.full_name}"
46
```

```
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     Codeium: Refactor | Explain
55 class LabTest(models.Model):
56     appointment = models.ForeignKey(Appointment, on_delete=models.CASCADE)
57     test_name = models.CharField(max_length=255)
58     description = models.TextField(blank=True, null=True)
59     result = models.TextField(blank=True, null=True)
60
61     Codeium: Refactor | Explain | Generate Docstring | X
62     def __str__(self):
63         return f"{self.test_name}"
64
65     Codeium: Refactor | Explain
66 class Prescription(models.Model):
67     appointment = models.ForeignKey(Appointment, on_delete=models.CASCADE)
68     medications = models.TextField(blank=True, null=True)
69
70     Codeium: Refactor | Explain | Generate Docstring | X
71     def __str__(self):
72         return f"Prescription for {self.appointment.patient.full_name}"
73
74     Codeium: Refactor | Explain
75 class Billing(models.Model):
76     patient = models.ForeignKey(patient_models.Patient, on_delete=models.SET_NULL,
77                                blank=True, null=True, related_name='billings')
78     appointment = models.ForeignKey(
79         Appointment, on_delete=models.CASCADE, related_name='billing', blank=True, null=True)
80     sub_total = models.DecimalField(max_digits=10, decimal_places=2)
81     tax = models.DecimalField(max_digits=10, decimal_places=2)
82     total = models.DecimalField(max_digits=10, decimal_places=2)
83     status = models.CharField(max_length=120, choices=[
84         ('Paid', 'Paid'), ('Unpaid', 'Unpaid')])
85     billing_id = ShortUUIDField(length=6, max_length=10, alphabet="1234567890")
86
87     date = models.DateTimeField(auto_now_add=True)
88
89     Codeium: Refactor | Explain | Generate Docstring | X
90     def __str__(self):
91         return f"Billing for {self.patient.full_name} - Total: {self.total}"
92
```

5. Update the ADMIN.PY

PATIENTS - ADMIN.PY

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

DOCTORS - ADMIN.PY

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

BASE - ADMIN.PY

```
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
10 class MedicalRecordInline(admin.TabularInline):
11     model = models.MedicalRecord
12     extra = 1
13
14 class LabTestInline(admin.TabularInline):
15     model = models.LabTest
16     extra = 1
17
18 class PrescriptionInline(admin.TabularInline):
19     model = models.Prescription
20     extra = 1
21
22 class BillingInline(admin.TabularInline):
23     model = models.Billing
24     extra = 1
25
26 class ServiceAdmin(ImportExportModelAdmin):
27     list_display = ['name', 'cost']
28     search_fields = ['name', 'description']
29     filter_horizontal = ['available_doctors']
30
31 class AppointmentAdmin(admin.ModelAdmin):
32     list_display = ['patient', 'doctor', 'appointment_date', 'status']
33     search_fields = ['patient_username', 'doctor_user_username']
34     inlines = [MedicalRecordInline, LabTestInline,
35               PrescriptionInline, BillingInline]
36
37 class MedicalRecordAdmin(admin.ModelAdmin):
38     list_display = ['appointment', 'diagnosis']
39
40 class LabTestAdmin(admin.ModelAdmin):
41     list_display = ['appointment', 'test_name']
42
43 class PrescriptionAdmin(admin.ModelAdmin):
44     list_display = ['appointment', 'medications']
45
46 class BillingAdmin(admin.ModelAdmin):
47     list_display = ['patient', 'total', 'status', 'date']
48
49 admin.site.register(models.Service, ServiceAdmin)
50 admin.site.register(models.Appointment, AppointmentAdmin)
51 admin.site.register(models.MedicalRecord, MedicalRecordAdmin)
52 admin.site.register(models.LabTest, LabTestAdmin)
53 admin.site.register(models.Prescription, PrescriptionAdmin)
54 admin.site.register(models.Billing, BillingAdmin)
```

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 image shows a VS Code editor window for a Django project named 'Health_Mgt_System'. The Explorer sidebar on the left shows the project structure, with the 'userauths' app and its 'migrations' folder highlighted in red. The main editor displays the 'admin.py' file for the 'userauths' app, containing code for three inline admin classes: 'AppointmentInline', 'MedicalRecordInline', and 'LabTestInline'. The terminal at the bottom shows the execution of three commands:

```
$ rm db.sqlite3 (venv)
Rosilie@DELL-MTNGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
$ python manage.py makemigrations userauths
Migrations for 'userauths':
  userauths/migrations/0001_initial.py
  - Create model User
(venv)
Rosilie@DELL-MTNGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
$ 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 APPMODELS.PY

```
HEALTH_MGT_SYSTEM base > models.py >
base
  > __pycache__
  > migrations
  > __init__.py
  > admin.py
  > apps.py
  > models.py
  > tests.py
  > views.py
  > doctor
  > HMS_main
  > patient
    > __pycache__
    > migrations
    > __init__.py
    > admin.py
    > apps.py
    > models.py
    > tests.py
    > views.py
  > Resource
  > static
  > templates
  > userauths
    > __pycache__
    > migrations
    > __pycache__
    > 0001_initial.py
    > __init__.py
    > admin.py
    > apps.py
    > 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    Codeium: Refactor | Explain | Generate Docstring | X
16    def __str__(self):
17        return f"{self.name} - {self.cost}"
18
19    Codeium: Refactor | Explain
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='service')
29     # doctor = models.ForeignKey(doctor_models.Doctor, on_delete=models.SET_NULL, null=True, blank=True, related_name='service')
30     # patient = models.ForeignKey(patient_models.Patient, on_delete=models.SET_NULL, null=True, blank=True, related_name='appointment')
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.

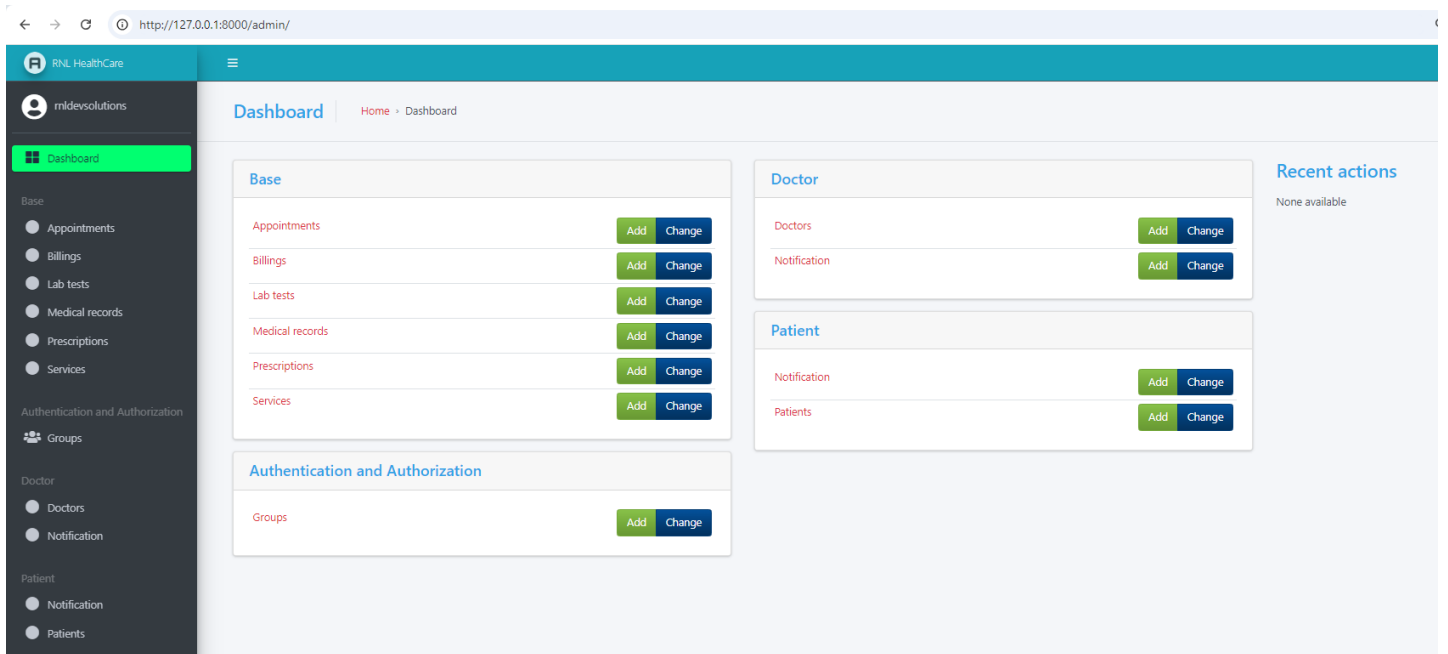
```
(venv)
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
• $ python manage.py makemigrations base
Migrations for 'base':
  base\migrations\0002_appointment_doctor_appointment_patient_and_more.py
  - Add field doctor to appointment
  - Add field patient to appointment
  - 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:



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

