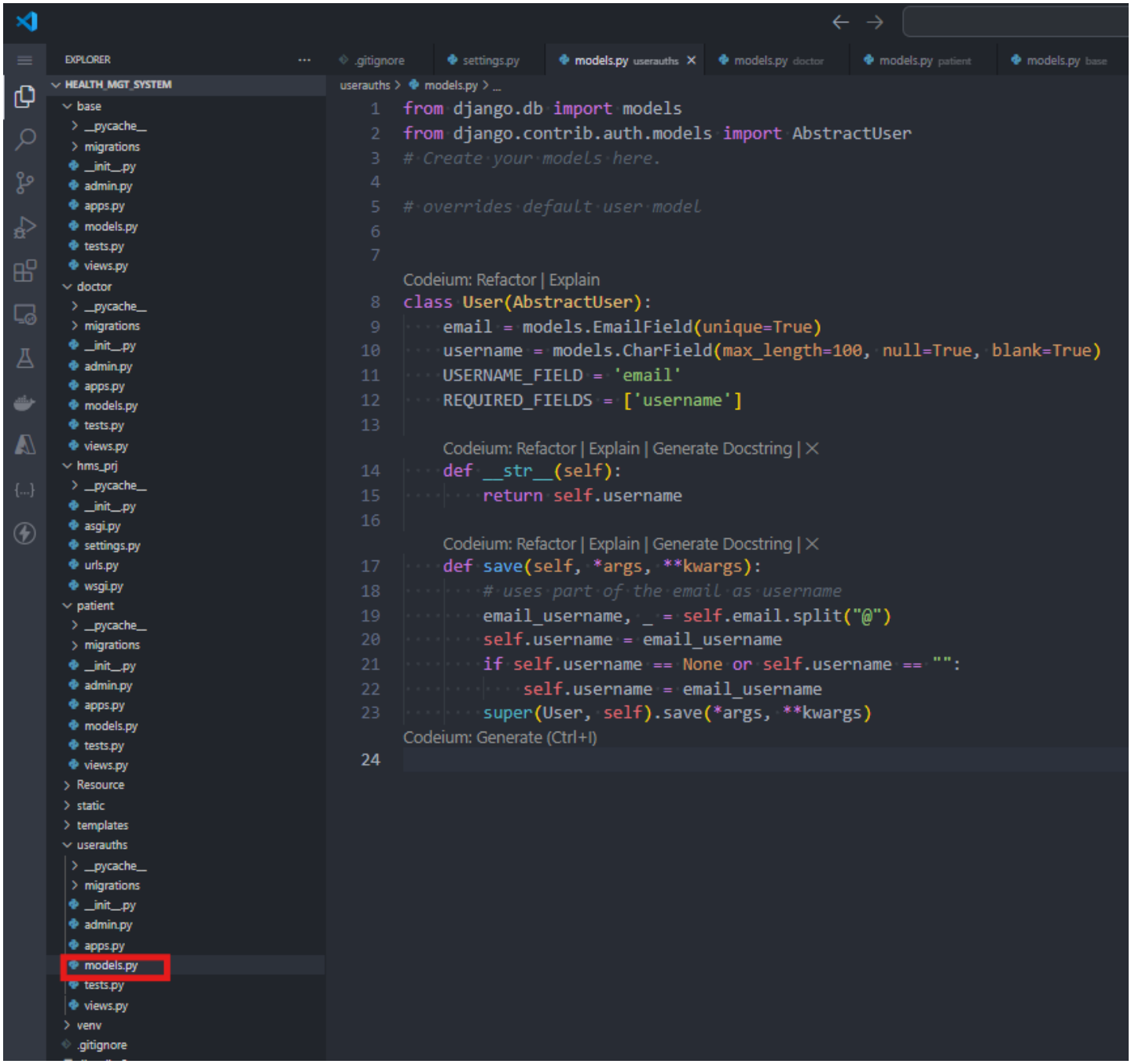
Topic: Models Part 2

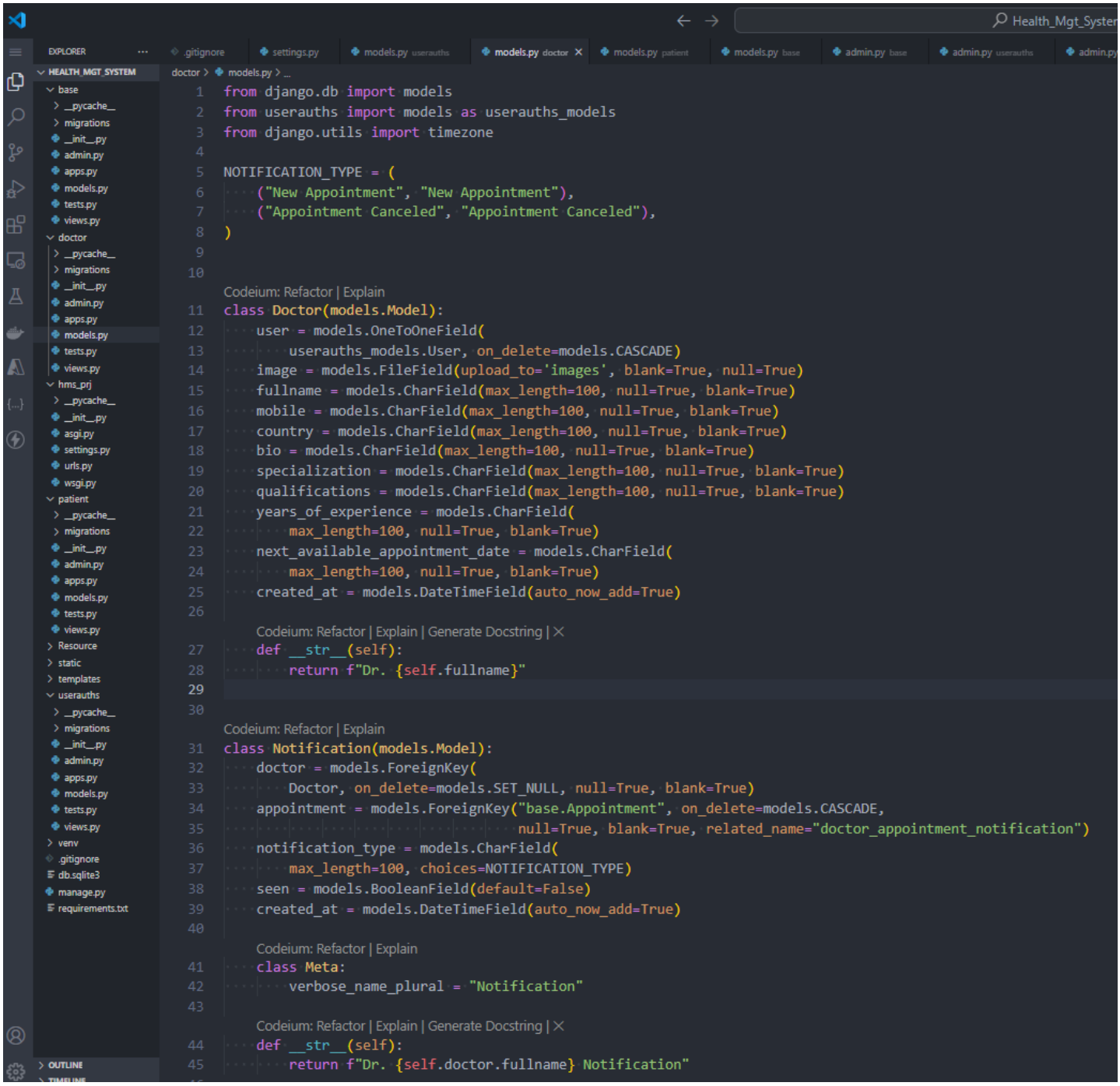
Speaker: | Notebook: Health Management System Using Django

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

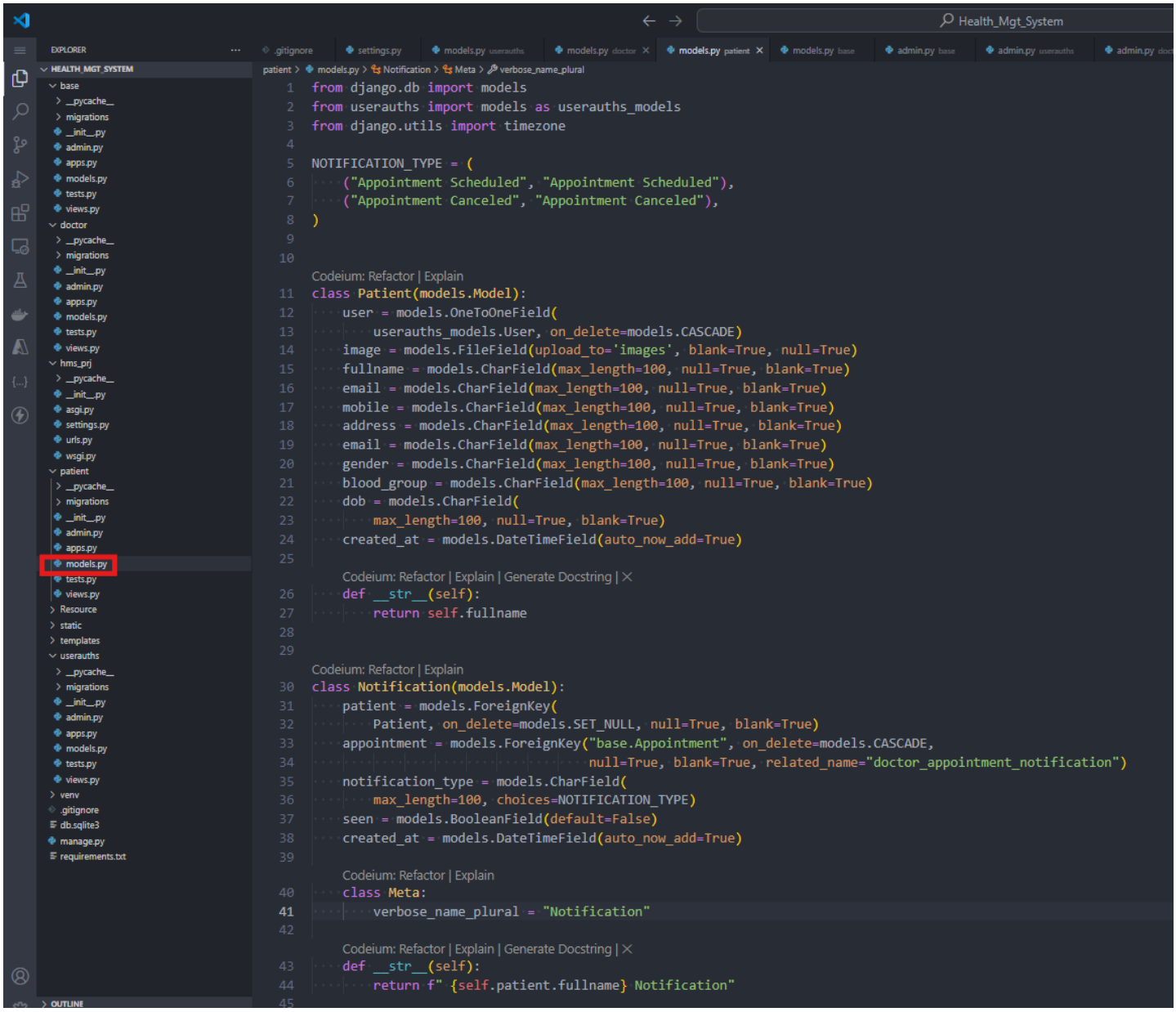
COMMON FIELDS HERE USE `'blank = True'` FOR THE PURPOSE OF IMPORTING THE DATA FROM CSV FILE, so as not to break our code.



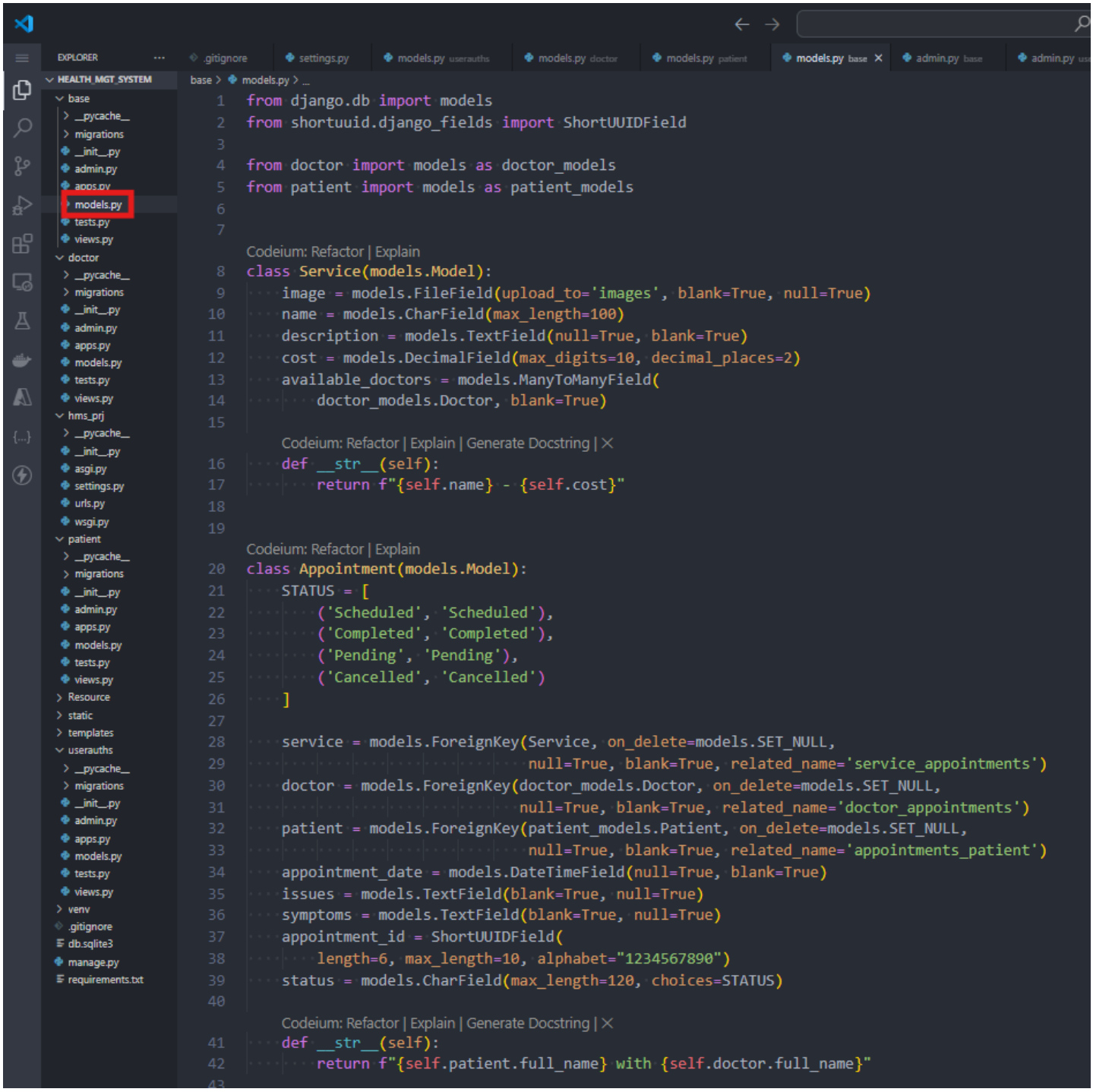
2. Update the doctor's app MODELS.PY

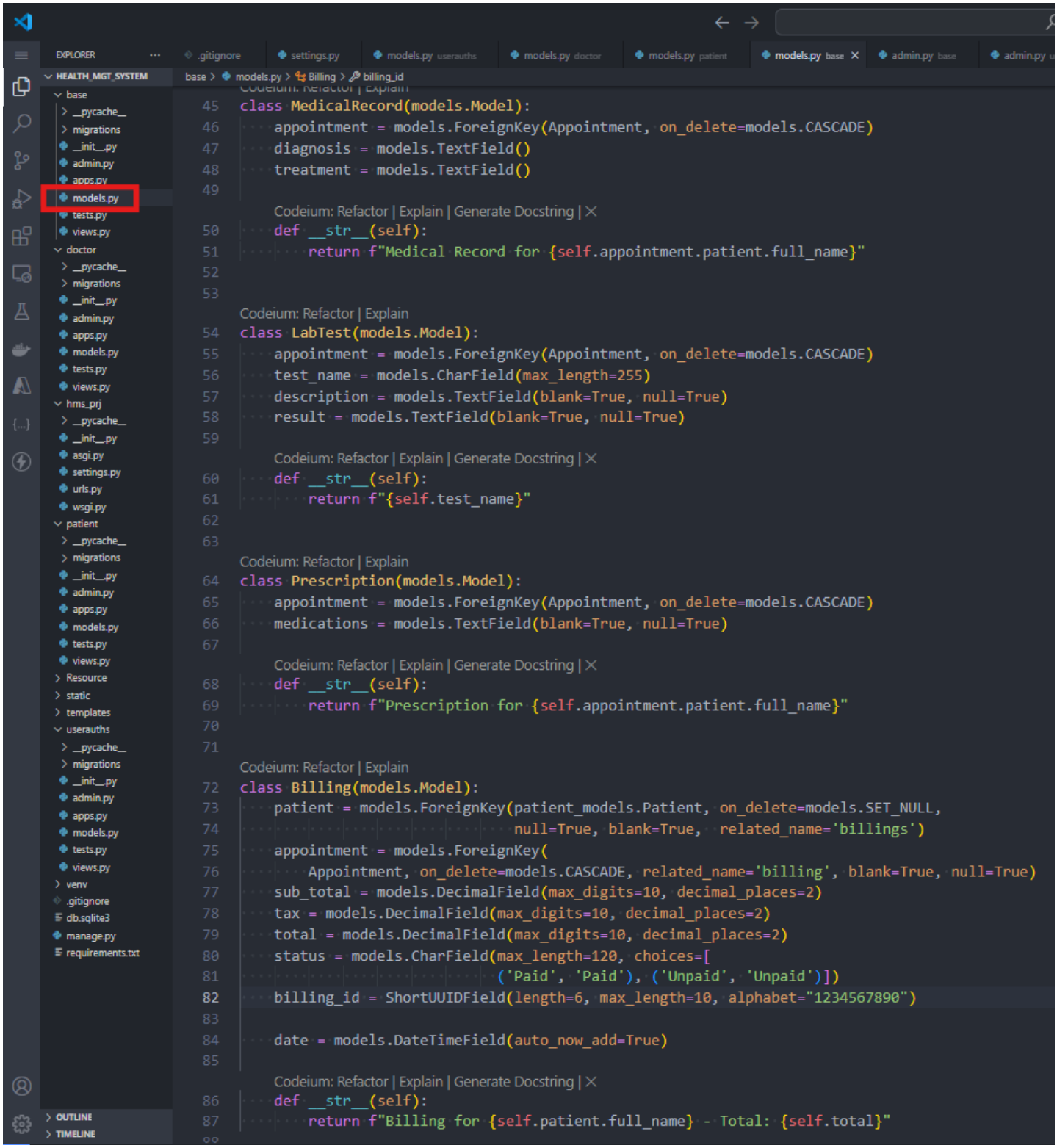


3. Update the patient's app MODELS.PY



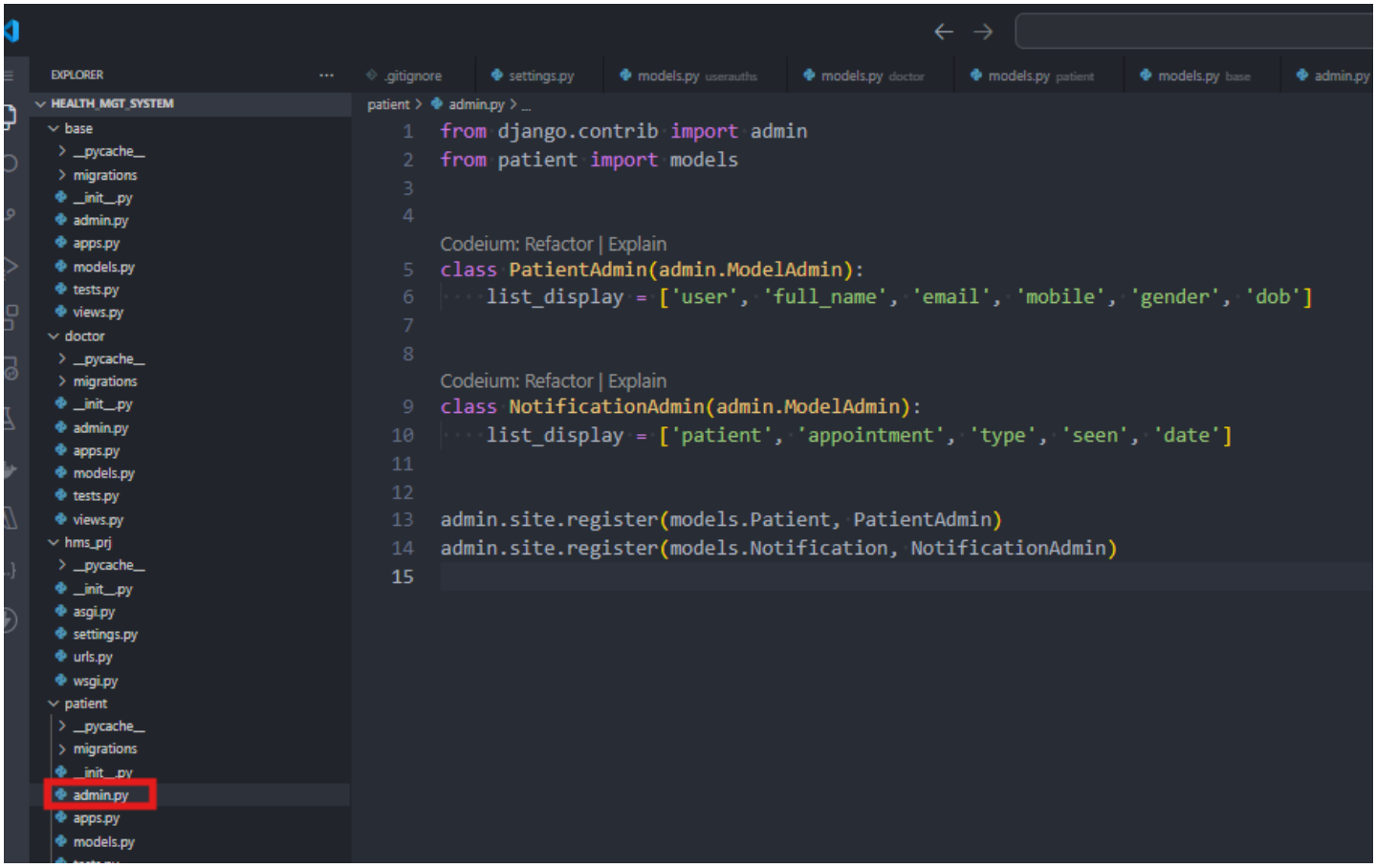
4. Update the base's app MODELS.PY


EXPLORER
HEALTH_MGT_SYSTEM
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_prj
__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
.gitignore
settings.py
models.py userauths
models.py doctor
models.py patient
models.py base
admin.py base
base > models.py > ...
from django.db import models
from shortuuid.django_fields import ShortUUIDField
from doctor import models as doctor_models
from patient import models as patient_models
Codeium: Refactor | Explain
class Service(models.Model):
image = models.FileField(upload_to='images', blank=True, null=True)
name = models.CharField(max_length=100)
description = models.TextField(null=True, blank=True)
cost = models.DecimalField(max_digits=10, decimal_places=2)
available_doctors = models.ManyToManyField(
doctor_models.Doctor, blank=True)
Codeium: Refactor | Explain | Generate Docstring |
def __str__(self):
return f"{self.name} - {self.cost}"
Codeium: Refactor | Explain
class Appointment(models.Model):
STATUS = [
('Scheduled', 'Scheduled'),
('Completed', 'Completed'),
('Pending', 'Pending'),
('Cancelled', 'Cancelled')
]
service = models.ForeignKey(Service, on_delete=models.SET_NULL,
null=True, blank=True, related_name='service_appointments')
doctor = models.ForeignKey(doctor_models.Doctor, on_delete=models.SET_NULL,
null=True, blank=True, related_name='doctor_appointments')
patient = models.ForeignKey(patient_models.Patient, on_delete=models.SET_NULL,
null=True, blank=True, related_name='appointments_patient')
appointment_date = models.DateTimeField(null=True, blank=True)
issues = models.TextField(blank=True, null=True)
symptoms = models.TextField(blank=True, null=True)
appointment_id = ShortUUIDField(
length=6, max_length=10, alphabet="1234567890")
status = models.CharField(max_length=120, choices=STATUS)
Codeium: Refactor | Explain | Generate Docstring |
def __str__(self):
return f"{self.patient.full_name} with {self.doctor.full_name}"

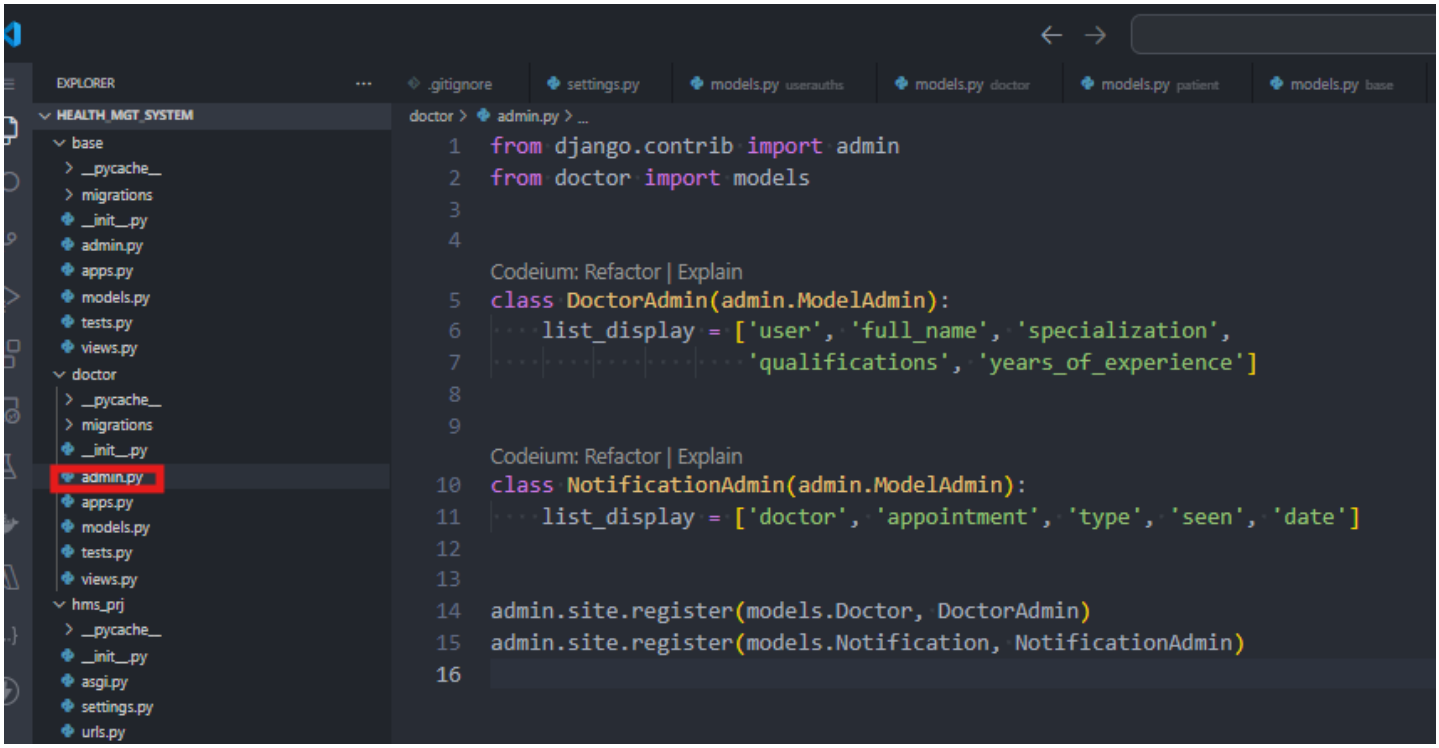


5. Update the ADMIN.PY

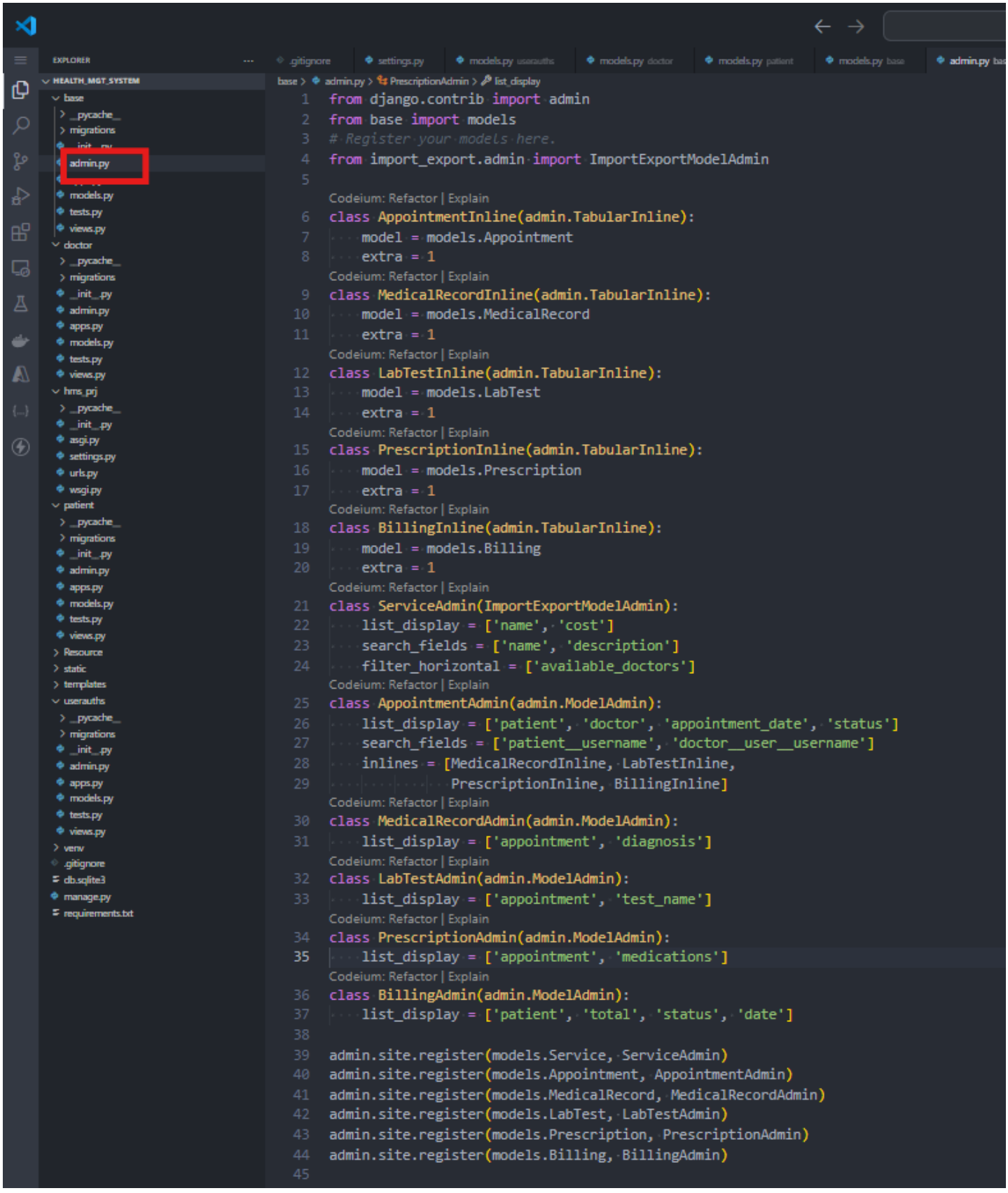
PATIENTS - ADMIN.PY



DOCTORS - ADMIN.PY



BASE - ADMIN.PY



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
rm db.sqlite3
```

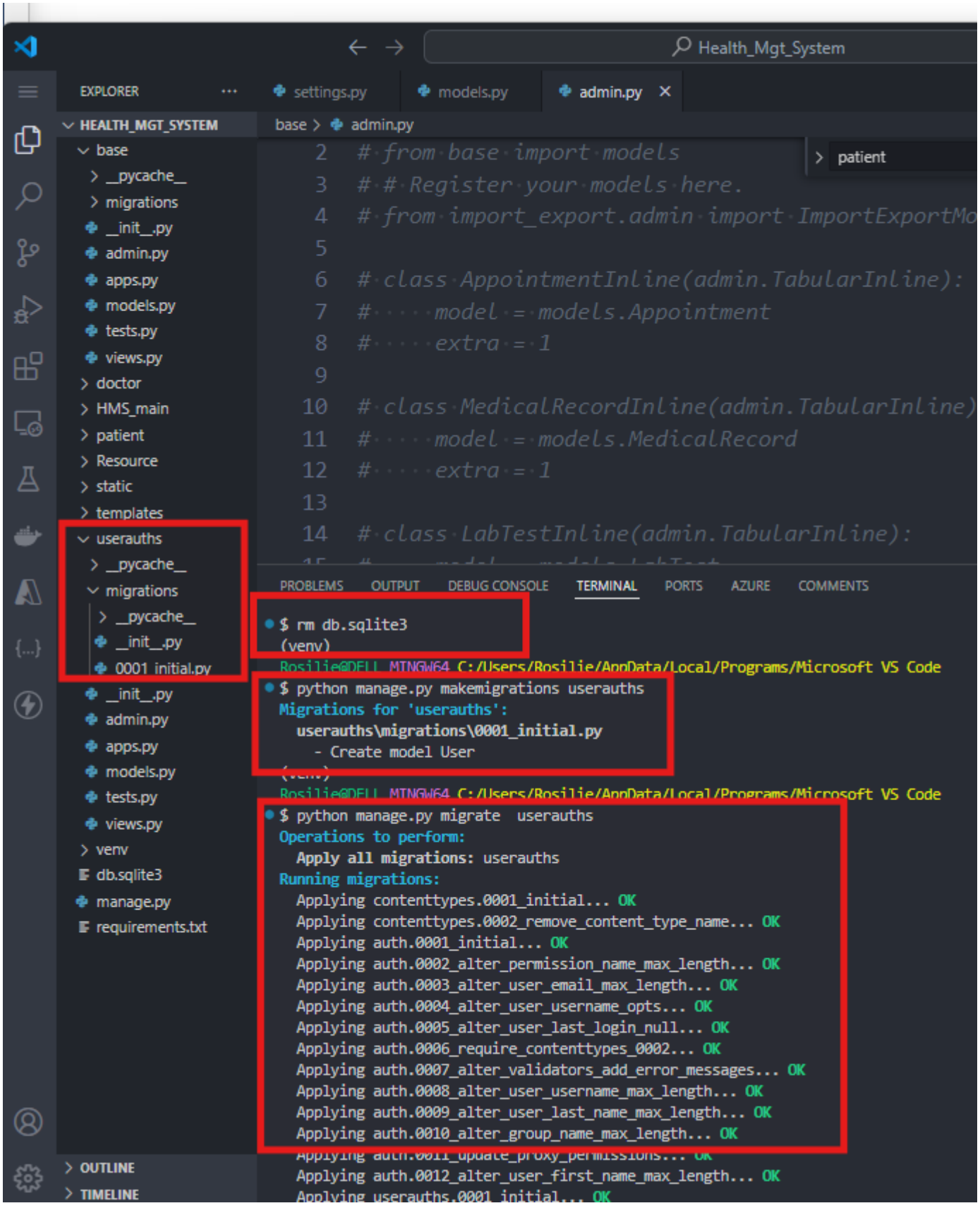
- **Delete migration files**: Go into each app's `migrations/` folder and delete all migration files except `__init__.py`.

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

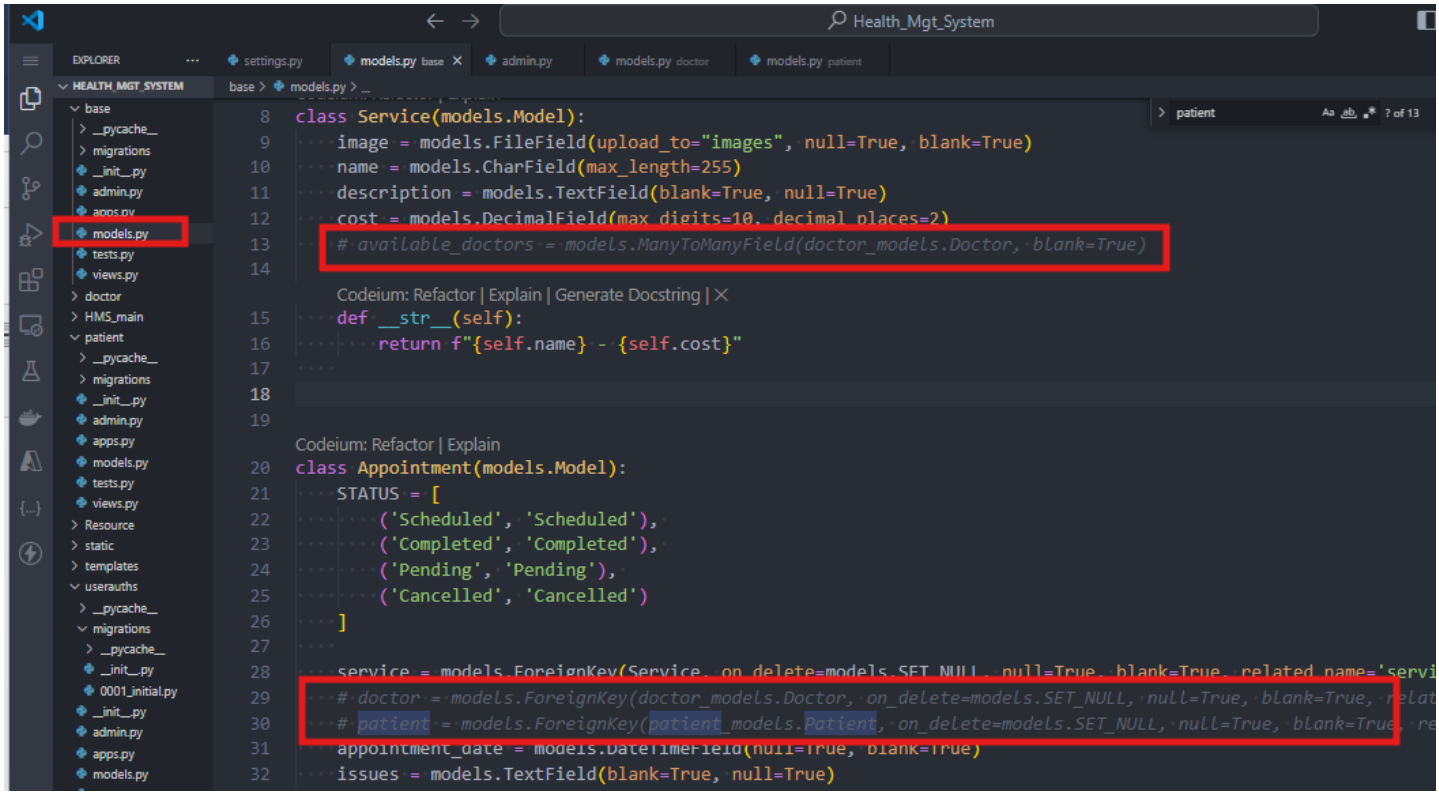
We run our migrations and USERAUTHS model works now. We then add our migrations on BASE app.



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

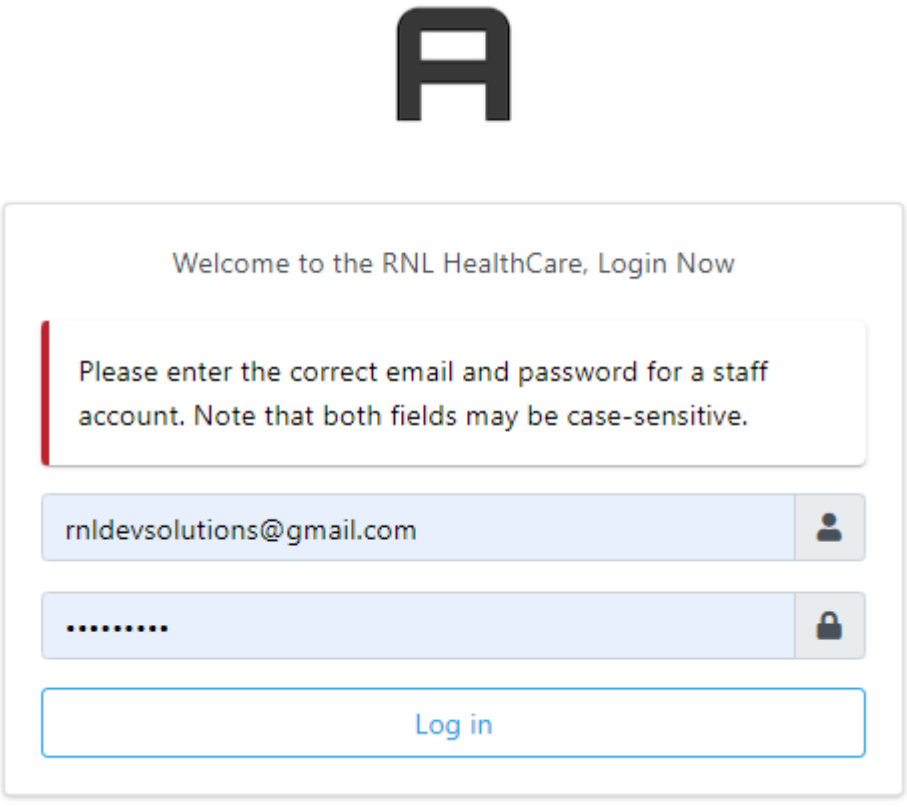


We then run the general migrations for all the default models of Django project.

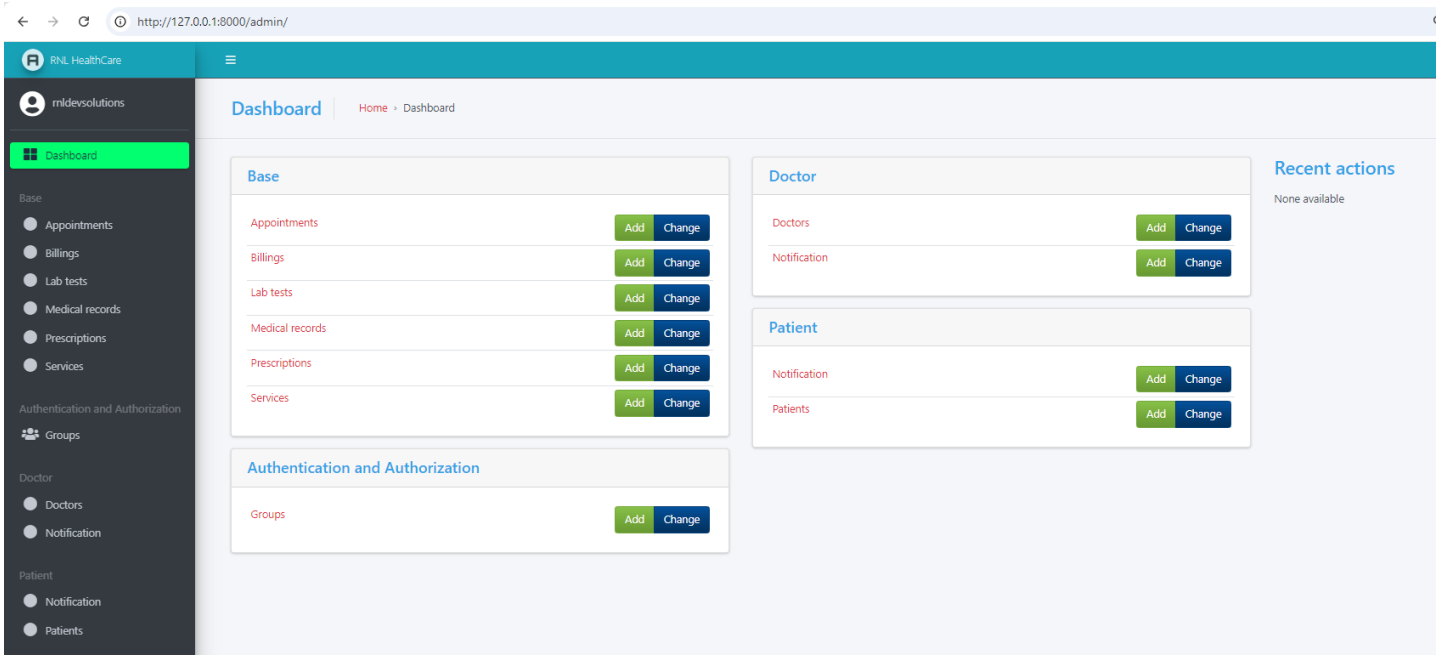
```
Rosilie@DELL MINGW64 C:/Users/Rosilie/AppData/Local/Programs/Microsoft VS Code
$ python manage.py makemigrations base
Migrations for 'base':
  base\migrations\0002_appointment_doctor_appointment_patient_and_more.py
    - Add field doctor to appointment
    - Add field patient to appointment
    -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



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