

## 1) MODELS

```
from django.db import models

class User(models.Model):
    ROLES = (
        ('admin', 'Admin'),
        ('student', 'Student'),
    )
    name = models.CharField(max_length=50)
    phone_number = models.CharField(max_length=15)
    address = models.CharField(max_length=60)
    role = models.CharField(max_length=10, choices=ROLES)
    password = models.CharField(max_length=25, default='Secret@123')

    def __str__(self):
        return self.name

class Course(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()

    # Many students can join many courses
    students = models.ManyToManyField(User, related_name='courses',
                                     blank=True)

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

`ManyToManyField` automatically creates a join table under the hood.

## Meaning of Each Parameter

Part	What it means
<code>ManyToManyField(User)</code>	Many courses can have many users (students), and one user can belong to many courses — so it's an M2M relationship.
<code>related_name='courses'</code>	Lets you access courses from the User model, like <code>user.courses.all()</code>
<code>blank=True</code>	Allows the field to be empty — a course can be created without students initially.

## 2) CRUD OPERATIONS — ASSOCIATE STUDENT TO COURSE

### A) Add student to a course

```
course = Course.objects.get(id=course_id)
student = User.objects.get(id=student_id, role='student')

course.students.add(student)
course.save()
```

### B) Remove student from a course

```
course = Course.objects.get(id=course_id)
student = User.objects.get(id=student_id)
course.students.remove(student)
```

### C) List students of a course

```
course = Course.objects.get(id=course_id)
students = course.students.all()
```

### D) List courses of a student

```
student = User.objects.get(id=student_id)
courses = student.courses.all() # because related_name='courses'
```

## E) Clear all students from course

```
course = Course.objects.get(id=course_id)
course.students.clear()
```

Many-to-Many from Course -> students, you can filter courses of students in two ways.

---

### 1) Get the student first, then list his courses

```
student = User.objects.get(id=student_id)
courses = student.courses.all()    # because of related_name='courses'
```

---

### 2) Filter directly on Course model using students\_\_id

```
courses = Course.objects.filter(students__id=student_id)
```

Both give you all courses that student is enrolled in.

Use **method 1** when you already have the student object.

Use **method 2** when you want a one-line queryset filter.