

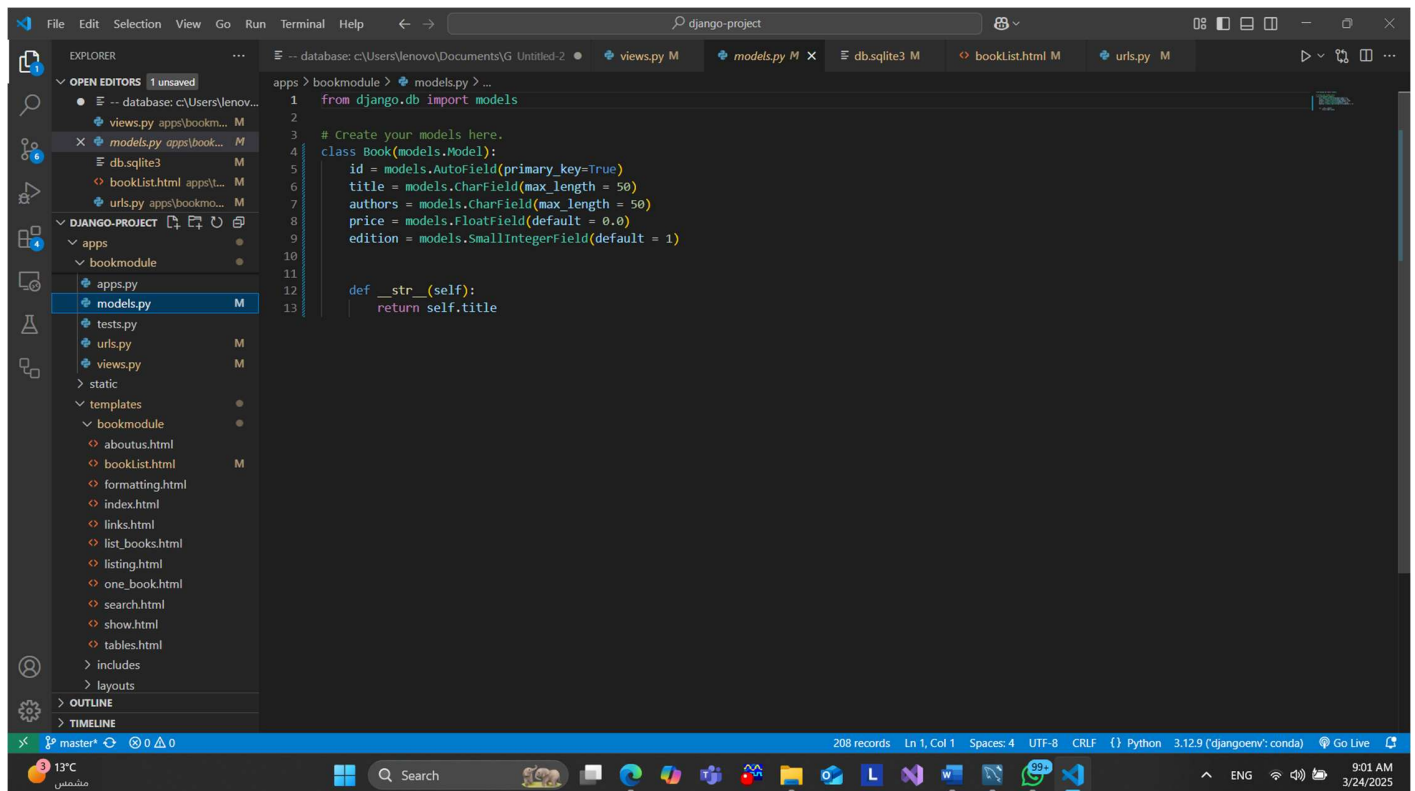
This lab session covers the basics of Django Models (Part 1). By the end of this session, students should have a very good understanding of how to build a basic model and how to access models in Django.

Pre-lab Preparation:

1. Having bookmodule and usermodule in apps directory, with necessary configurations in settings.py.

Lab Activities:

Task 1: using models, we create a database scheme

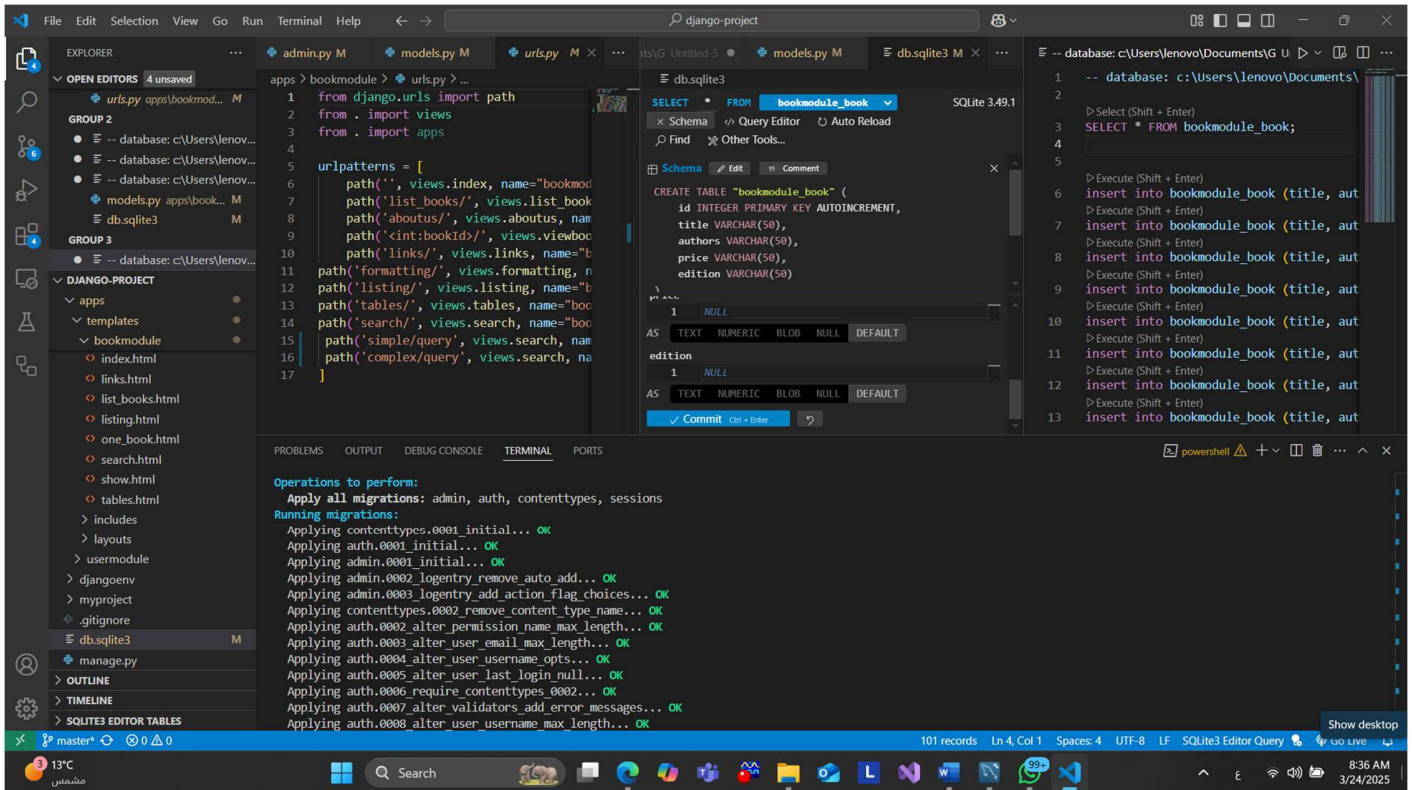


The screenshot shows a VS Code editor with a Django project named 'django-project'. The Explorer panel on the left shows the project structure, including the 'apps' directory with 'bookmodule' and 'usermodule' subdirectories. The 'models.py' file in 'bookmodule' is open in the editor. The code in 'models.py' defines a 'Book' model with the following fields:

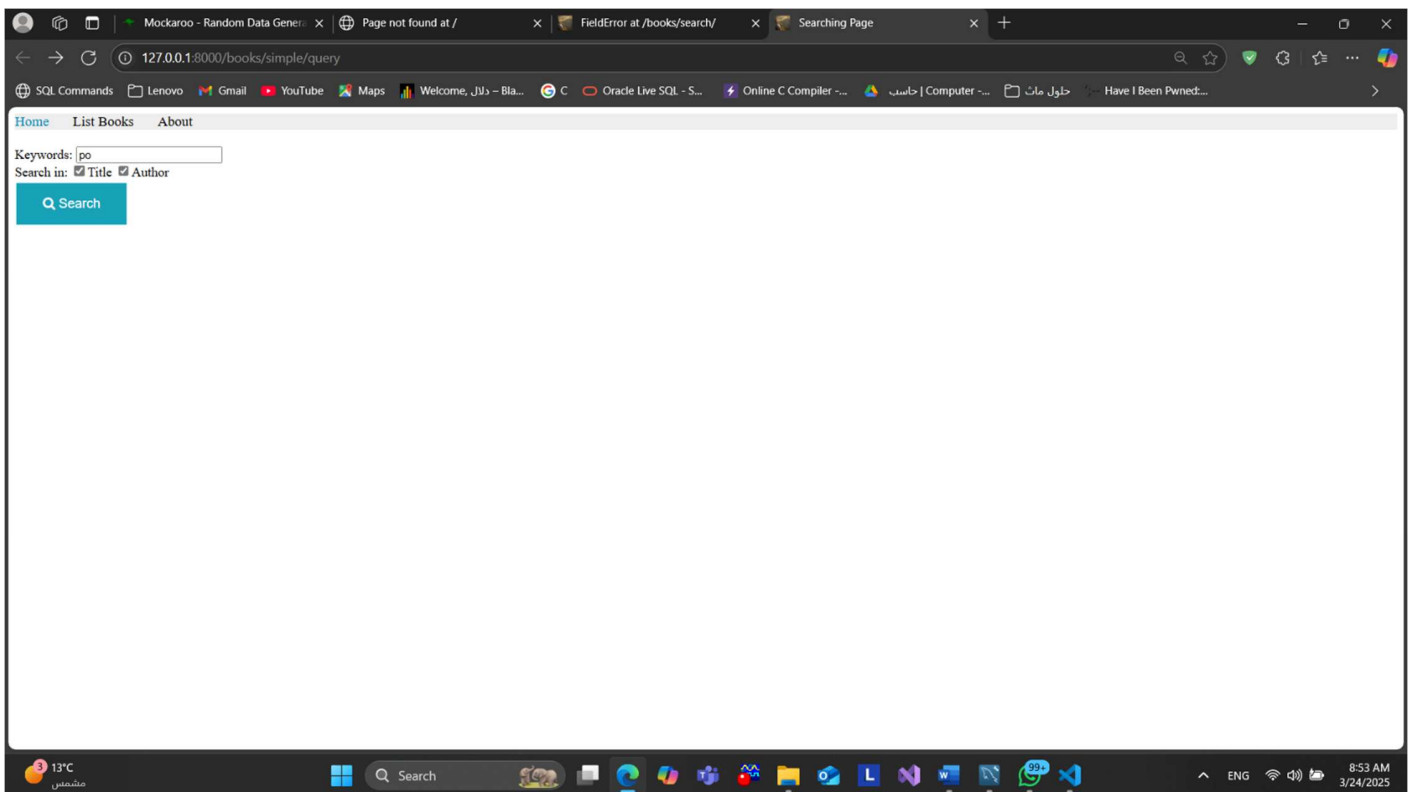
```
1 from django.db import models
2
3 # Create your models here.
4 class Book(models.Model):
5     id = models.AutoField(primary_key=True)
6     title = models.CharField(max_length = 50)
7     authors = models.CharField(max_length = 50)
8     price = models.FloatField(default = 0.0)
9     edition = models.SmallIntegerField(default = 1)
10
11
12 def __str__(self):
13     return self.title
```

Task 2: use the application 'DB Browser for SQLite' or 'VS code' to view the database and insert fake data into it.

Note: For VS code, you need *SQLite extension*, and for SQLite, you will use *sqlitebrowser* at (<https://sqlitebrowser.org/dl/>). In the lab, we assumed that you are familiar with at least one of them.



Task 3: Use simple queries using Django ORM.



The screenshot displays a web browser window at the top and a code editor (VS Code) at the bottom, both showing the same Django application.

Browser Window:

- Address bar: `127.0.0.1:8000/books/complex/query`
- Page content: A list of book titles and their IDs, followed by a search form.
- Search results: A table showing the results of the search query.

Code Editor (VS Code):

- File Explorer: Shows the project structure, including `apps`, `bookmodule`, `templates`, and `static` directories.
- Code Editor: Displays the `views.py` file, which contains the `search` and `__getBooksList` functions.

Code Snippets:

```
def search(request):
    if request.method == "POST":
        keyword = request.POST.get('keyword', '').strip().lower()
        is_title = request.POST.get('option1')
        is_author = request.POST.get('option2')

        books = Book.objects.all()

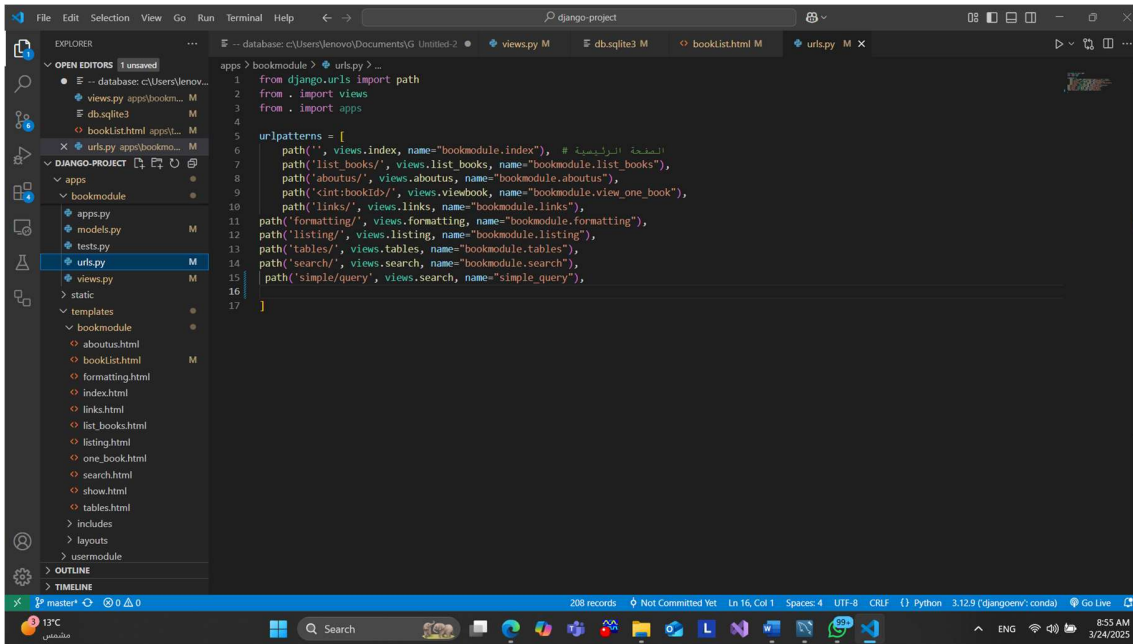
        if keyword:
            if is_title and is_author:
                books = books.filter(
                    models.Q(title__icontains=keyword) | models.Q(author__icontains=keyword)
                )
            elif is_title:
                books = books.filter(title__icontains=keyword)
            elif is_author:
                books = books.filter(author__icontains=keyword)
            else:
                books = []

        return render(request, 'bookmodule/bookList.html', {'books': books})
    return render(request, 'bookmodule/search.html')
```

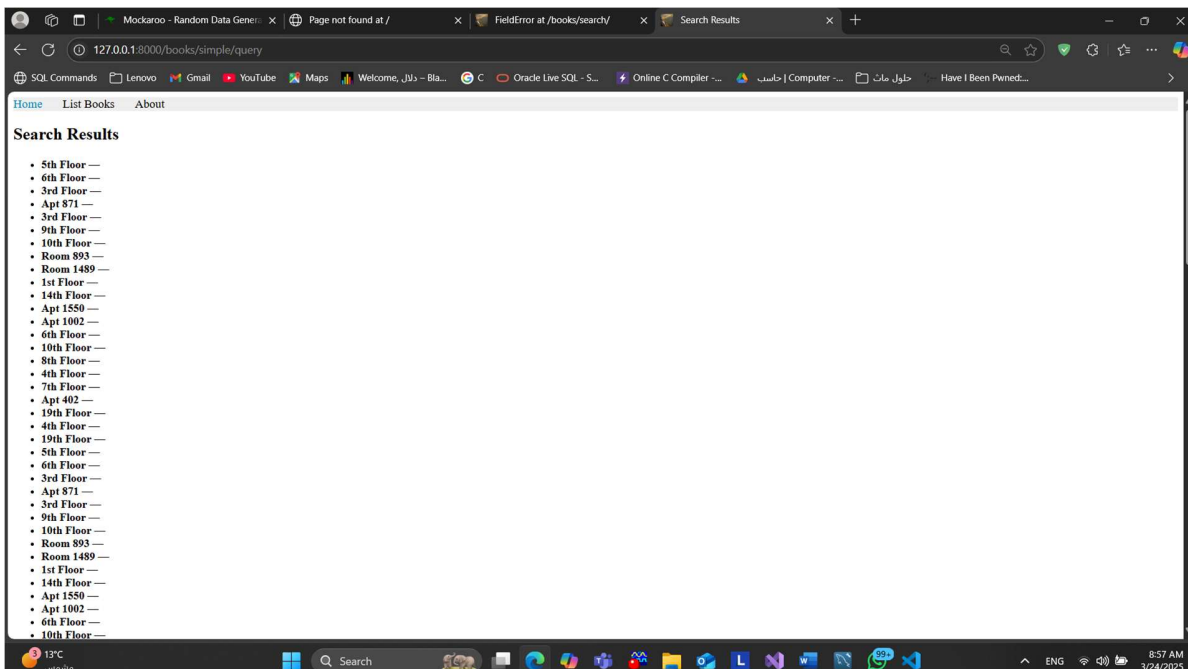
```
def __getBooksList():
    book1 = {'id': 12344321, 'title': 'Continuous Delivery', 'author': 'J. Humble and D. Farley'}
    book2 = {'id': 56788765, 'title': 'Reversing: Secrets of Reverse Engineering', 'author': 'E. Eilam'}
    book3 = {'id': 43211234, 'title': 'The Hundred-Page Machine Learning Book', 'author': 'Andriy Burkov'}
    return [book1, book2, book3]

mybook = Book(title='Continuous Delivery', authors='J. Humble and D. Farley', edition=1)
mybook.save()

def simple_query(request):
    mybooks = Book.objects.filter(title__icontains='and') # <- multiple objects
    return render(request, 'bookmodule/bookList.html', {'books': mybooks})
```



```
1 from django.urls import path
2 from . import views
3 from . import apps
4
5 urlpatterns = [
6     path('', views.index, name="bookmodule.index"), # الصفحة الرئيسية
7     path('list_books/', views.list_books, name="bookmodule.list_books"),
8     path('aboutus/', views.aboutus, name="bookmodule.aboutus"),
9     path('dint:bookidb/', views.viewbook, name="bookmodule.view_one_book"),
10    path('links/', views.links, name="bookmodule.links"),
11    path('formatting/', views.formatting, name="bookmodule.formatting"),
12    path('listing/', views.listing, name="bookmodule.listing"),
13    path('tables/', views.tables, name="bookmodule.tables"),
14    path('search/', views.search, name="bookmodule.search"),
15    path('simple/query', views.search, name="simple_query"),
16
17 ]
```



The screenshot displays a web browser window at the top and a code editor (VS Code) at the bottom. The browser shows a list of book entries with IDs and titles. The code editor shows the Django project structure and the views.py file.

Browser Window:

- Address bar: 127.0.0.1:8000/books/simple/query
- Search bar: Search
- Results:

 - ID: 24, Title: 9th Floor
 - ID: 25, Title: 10th Floor
 - ID: 26, Title: Room 893
 - ID: 35, Title: Room 1489
 - ID: 38, Title: 1st Floor
 - ID: 47, Title: 14th Floor
 - ID: 58, Title: Apt 1550
 - ID: 61, Title: Apt 1002
 - ID: 71, Title: 6th Floor
 - ID: 78, Title: 10th Floor
 - ID: 80, Title: 8th Floor
 - ID: 83, Title: 4th Floor
 - ID: 84, Title: 7th Floor
 - ID: 85, Title: Apt 402
 - ID: 86, Title: 19th Floor
 - ID: 93, Title: 4th Floor
 - ID: 96, Title: 19th Floor
 - ID: 106, Title: 5th Floor
 - ID: 112, Title: 6th Floor
 - ID: 116, Title: 3rd Floor
 - ID: 117, Title: Apt 871

Code Editor (VS Code):

- File Explorer: Shows the project structure with folders like static, templates, bookmodule, and files like apps.py, models.py, tests.py, urls.py, views.py.
- Search Bar: Search
- Code Editor: Shows the views.py file with the following code:

```
def complex_query(request):
    mybooks=books=Book.objects.filter(author__isnull = False).filter(title__icontains='and').filter(edition__gte = 2).exclude(price__lte = 1)
    if len(mybooks)>=1:
        return render(request, 'bookmodule/booklist.html', {'books':mybooks})
    else:
        return render(request, 'bookmodule/index.html')
```

CS471 – Web Technologies (Laboratory)		Lab Week 7
		Django Models (Part 1)