



جامعة الازقية
كلية الهندسة المعلوماتية
سنة رابعة برمجيات
المادة: تحليل نظم مالية

تحليل نظام تعلم ذكي Intelligent Learning System

بإشراف:

الدكتور غيث بلال
الانسة زينب محفوض

إعداد الطلاب:

يزن هشام متوج
نغم نزار عثمان
بشير اسماعيل خيربك
حسن زياد قناية

تحليل نظام تعلم ذكي Intelligent Learning System

لدنيا نظام " تعلم بذكاء " هو صفة تعليمية الكترونية تهدف إلى تقديم محتوى تعليمي متنوع للطلاب من مختلف المستويات. يتيح النظام للطلاب الوصول إلى الدروس والاختبارات والموارد التعليمية والتفاعل مع المعلمين وزملائهم.

حيث يزور الطالب الموقع وينقر على زر "تسجيل الدخول" و يدخل اسم المستخدم وكلمة المرور، فإذا كانت المعلومات صحيحة يتم توجيه الطالب إلى الصفحة الرئيسية للنظام، بعد تسجيل الدخول ، يظهر للطلاب قائمة بالدورات المتاحة إذ يمكنه تصفح الدورات حسب الفئة أو البحث عن دورة معينة، عند النقر على دورة معينة يتم عرض تفاصيل الدورة (الوصف المحتوى ، المعلم ، ...) ، يسجل الطالب على الدورة من خلال اختيار دورة والضغط على زر " التسجيل " ، ويتم إضافة الدورة إلى قائمة الدورات الخاصة بالطالب ، يدخل الطالب إلى صفحة الدورة المسجلة ويمكنه الوصول إلى المحتوى التعليمي (فيديوهات ، مستندات ، ملاحظات) و يمكنه تحميل الموارد المتاحة ، يكون للطالب خيار إرسال رسالة إلى المعلم عبر نظام الرسائل داخل المنصة، فيتلقى المعلم الاشعار ويرد على استفسارات الطلاب.

بعد انتهاء الطالب من دراسة المحتوى يمكنه إجراء اختبار نهاية الدورة، حيث يتم الاختبار بشكل تفاعلي مع خيارات متعددة، بعد انتهاء الاختبار تظهر للطالب نتيجة الاختبار، يمكن للطالب الوصول إلى صفحة "تقدمي" لرؤية الأداء في الدورات المختلفة (تتبع التقدم) حيث يتم عرض الرسوم البيانية التي توضح نسبة الانجاز والدرجات، يقوم المعلم بإدارة الدورات، حيث أنه يقوم بتسجيل الدخول إلى حسابه والذهاب إلى لوحة التحكم، ويمكنه أيضاً إنشاء دورة جديدة وإضافة محتوى وتحديد مواعيد الاختبارات، ويقوم المعلم بمراجعة نتائج الطلاب وإرسال الملاحظات. الإداريين يقومون بإدارة النظام حيث يقوم الإداري بالدخول إلى النظام وإدارة المستخدمين (إضافة- حذف طلاب والمعلمين) ويمكنه مراقبة الأداء العام للدورات وتقديم تقارير عن التقدم.

المستخدمون والممثلون:

- الطالب:** يستخدم النظام للوصول للدورات التعليمية والاختبارات.
- المعلم:** يقوم بإنشاء وإدارة الدورات والاختبارات.
- الإداري:** يدير المستخدمين ويراقب الأداء العام للنظام.

حالات الاستخدام

تسجيل دخول الطالب: الطالب يدخل اسم المستخدم وكلمة المرور، ثم يتم توجيهه إلى الصفحة الرئيسية أو تظهر رسالة خطأ.

استعراض الدورات والتسجيل: الطالب يتصفح الدورات حسب الفئة أو البحث، ثم يسجل في دورة معينة.

مشاهدة محتوى الدورة: الطالب يدخل إلى الدورة ويشاهد الفيديوهات والوثائق ويستطيع تحميل الموارد.

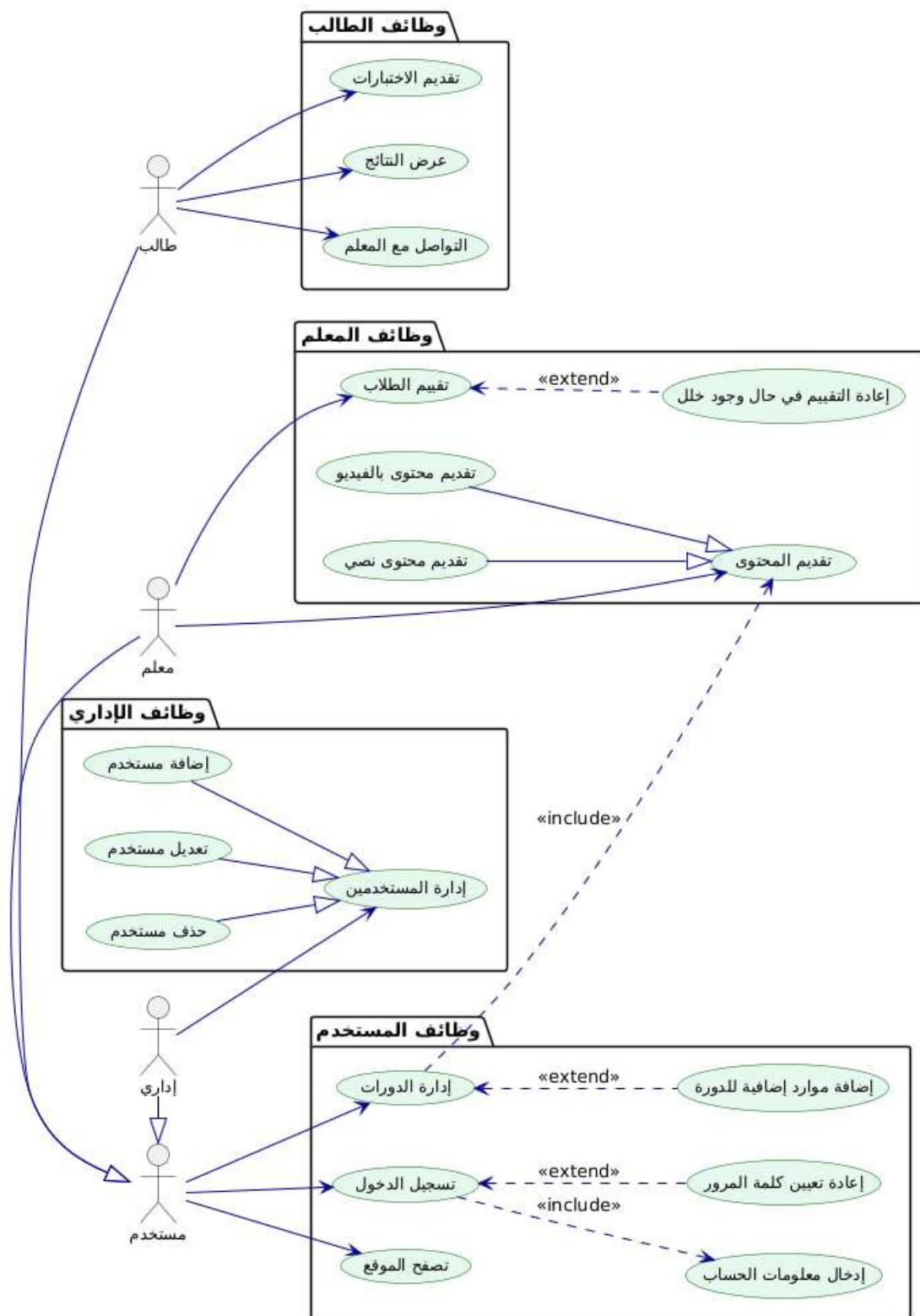
أداء اختبار الدورة: الطالب يبدأ الاختبار التفاعلي ويحصل على النتيجة فور الانتهاء.

إرسال رسالة إلى المعلم: الطالب يرسل رسالة عبر النظام ويتلقى رد من المعلم.

إنشاء دورة جديدة: المعلم يدخل لوحة التحكم وينشئ دورة جديدة ويضيف محتوى واختبارات.

إدارة المستخدمين: الإداري يضيف أو يحذف المستخدمين من النظام

والان لدينا مخطط حالة الاستخدام: UseCase Digram



المتطلبات الوظيفية:

الوصف	المتطلب
تمكين المستخدمين من تسجيل الدخول باستخدام اسم المستخدم وكلمة المرور.	تسجيل دخول المستخدم
عرض قائمة الدورات التعليمية حسب الفئة أو من خلال البحث	استعراض الدورات
تمكين الطالب من التسجيل في الدورات التعليمية	التسجيل في دورة
تمكين الطالب من الوصول إلى الفيديوهات، المستندات، والملاحظات الخاصة بالدورة	مشاهدة محتوى الدورة
تمكين الطالب من تحميل الموارد المتاحة في الدورة	تحميل الموارد
تمكين الطلاب والمعلمين من إرسال واستقبال الرسائل داخل المنصة	المراسلة الداخلية
توفير اختبار تفاعلي بنمط الاختيارات المتعددة للطلاب	أداء اختبار الدورة
عرض نتائج الاختبارات فور الانتهاء منها	عرض نتيجة الاختبار
عرض نسبة الإنجاز والدرجات في صفحة مخصصة للطالب	تتبع التقدم
تمكين المعلم من إنشاء الدورات وتحديد مواعيد الاختبارات	إدارة الدورات
تمكين المعلم من مراجعة نتائج الطلاب وإرسال الملاحظات	مراجعة النتائج
تمكين الإداري من إضافة أو حذف المستخدمين	إدارة المستخدمين
توفير تقارير عن الأداء والتقدم للمدير	تقارير النظام

المتطلبات غير الوظيفية:

الوصف	المتطلب
يجب أن يكون النظام متاحًا 7/24 بنسبة تشغيل لا تقل عن 99%	التوفر
تشفير بيانات الدخول واستخدام JWT للجلسات	الأمان
زمن استجابة العمليات لا يتجاوز 1.5 ثانية	الأداء
دعم الهواتف والحواسيب باستخدام تصميم متجاوب	التوافق
دعم آلاف المستخدمين المتزامنين مستقبلاً.	التوسع

الان سنبدأ بتوصيف حالتي استخدام:

الحالة الأولى: تسجيل دخول المستخدم:

الوصف:

يتيح للطالب مشاهدة مقاطع الفيديو، تحميل المستندات والملاحظات الخاصة بالدورة.

المحفز:

زيارة صفحة الدخول في الموقع.

الجهات الفاعلة:

- الممثل الرئيسي: الطالب
- الممثل الثانوي: النظام (خادم المصادقة)

الشرط المسبق:

الطالب مسجل في الدورة

الشرط اللاحق:

الطالب يتمكن من تصفح و/أو تحميل المحتوى

. حالات الخطأ:

1. لم يتم تسجيل الطالب في الدورة
2. المحتوى غير متاح أو مفقود
3. فشل تحميل الملفات

تصرف النظام عند الخطأ:

يعرض رسالة: "يجب التسجيل في الدورة للوصول للمحتوى"
في حال فقدان الملف: "المورد غير متوفر حالياً"
في حال فشل التحميل: "حدث خطأ أثناء التحميل، يرجى المحاولة مرة أخرى"

العمليات الأساسية:

1. الطالب يفتح صفحة تسجيل الدخول.
2. يُدخل اسم المستخدم وكلمة المرور.
3. النظام يتحقق من صحة البيانات.
4. إذا كانت صحيحة، يتم توجيه الطالب إلى الصفحة الرئيسية

العمليات البديلة:

- . إذا كان المستخدم "معلماً" أو "إدارياً"، يتم تحويله إلى لوحة تحكم خاصة به بحسب الدور.
- . خيار "نسيت كلمة المرور؟" يمكن استخدامه لإعادة تعيينها

الحالة الثانية: أداء اختبار الدورة:

الوصف:

يسمح للطالب بإجراء اختبار بعد إنهاء محتوى الدورة وتقييم أدائه.

الشرط المسبق:

الطالب أنهى محتوى الدورة، والدورة مفعلة للاختبار.

الشرط اللاحق:

يتم حفظ إجابات الطالب وتُعرض النتيجة فورًا.

. حالات الخطأ:

1. الطالب يحاول إجراء الاختبار قبل إتمام المحتوى
2. فقد الاتصال أثناء الحل
3. إرسال غير مكتمل للإجابات

تصرف النظام عند حدوث خطأ:

1. إذا لم يُكمل الطالب محتوى الدورة:
 - يعرض النظام رسالة مثل:
"يجب إكمال محتوى الدورة قبل إجراء الاختبار".
2. إذا انقطع الاتصال أثناء الحل:
 - يحفظ النظام الإجابات تلقائيًا حتى آخر نقطة.
 - يعرض رسالة:
"تم فقد الاتصال مؤقتًا. جاري المحاولة لإعادة الاتصال... لا تغلق الصفحة".

3. إذا كانت الإجابات غير مكتملة عند الإرسال:

◦ يمنع الإرسال ويُظهر تنبيه:

"الرجاء الإجابة على جميع الأسئلة قبل إرسال الاختبار".

4. في حال فشل الحفظ النهائي للإجابات (خطأ خادم):

◦ يعرض رسالة:

"حدث خطأ أثناء إرسال الاختبار. لم يتم تسجيل إجاباتك. الرجاء إعادة المحاولة".

◦ يتم تسجيل الخطأ في سجل النظام ليتمكن الدعم من مراجعته

العمليات الأساسية:

1. الطالب ينهي المحتوى التدريسي للدورة.
2. يضغط على "ابدأ الاختبار".
3. يظهر له نموذج الأسئلة (خيارات متعددة).
4. الطالب يُجيب على الأسئلة.
5. ينقر على "إرسال".
6. النظام يصحّح الإجابات تلقائيًا ويعرض النتيجة.

العمليات البديلة:

- يمكن للطالب حفظ التقدم مؤقتًا واستئناف الاختبار لاحقًا (في حال توفر هذه الميزة).
- النظام يعطي ملاحظات لكل سؤال بعد التقييم (إذا كانت مفعلة)

الآن سنبدأ برسم مخطط الصفوف Class Digram:

المؤلف من:

أولاً: الكيانات (Entities) :

1. الطالب (Student):

يمثل المستخدم الذي يتلقى المحتوى التعليمي ويؤدي الاختبارات

2. المعلم (Instructor):

يقوم بإنشاء الدورات، إضافة المحتوى، وتصحيح الاختبارات.

3. الدورة التعليمية (Course) :

تمثل المحتوى التعليمي الذي يسجل فيه الطلاب.

4. المحتوى (Content) :

يشمل الفيديوهات، الملفات، والملاحظات ضمن الدورة.

5. الاختبار (Exam):

يحتوي على أسئلة نهاية الدورة.

6. الإجابة والنتيجة (Result) :

تمثل أداء الطالب في اختبار معين.

7. الرسائل (Message):

نظام تواصل داخلي بين الطالب والمعلم.

8. الإداري (Admin):

يدير المستخدمين ويتابع تقارير النظام.

ثانياً: العلاقات (Relationships) :

1. Student → Course (N إلى N)

الطلاب يمكنه التسجيل في عدة دورات، وكل دورة يمكن أن يلتحق بها عدة طلاب. يتم تمثيل هذه العلاقة باستخدام جدول وسيط يسمى "Enrollment".

2. Course → Instructor (1 إلى N)

كل دورة تعليمية يشرف عليها معلم واحد، ولكن المعلم يمكنه الإشراف على أكثر من دورة.

3. Course → Content (1 إلى N)

كل دورة تحتوي على عدة وحدات من المحتوى مثل فيديوهات، مستندات، وملاحظات.

4. Course → Exam (1 إلى 1)

لكل دورة اختبار واحد نهائي يتم ربطه بها مباشرة.

5. Exam & Student → Result

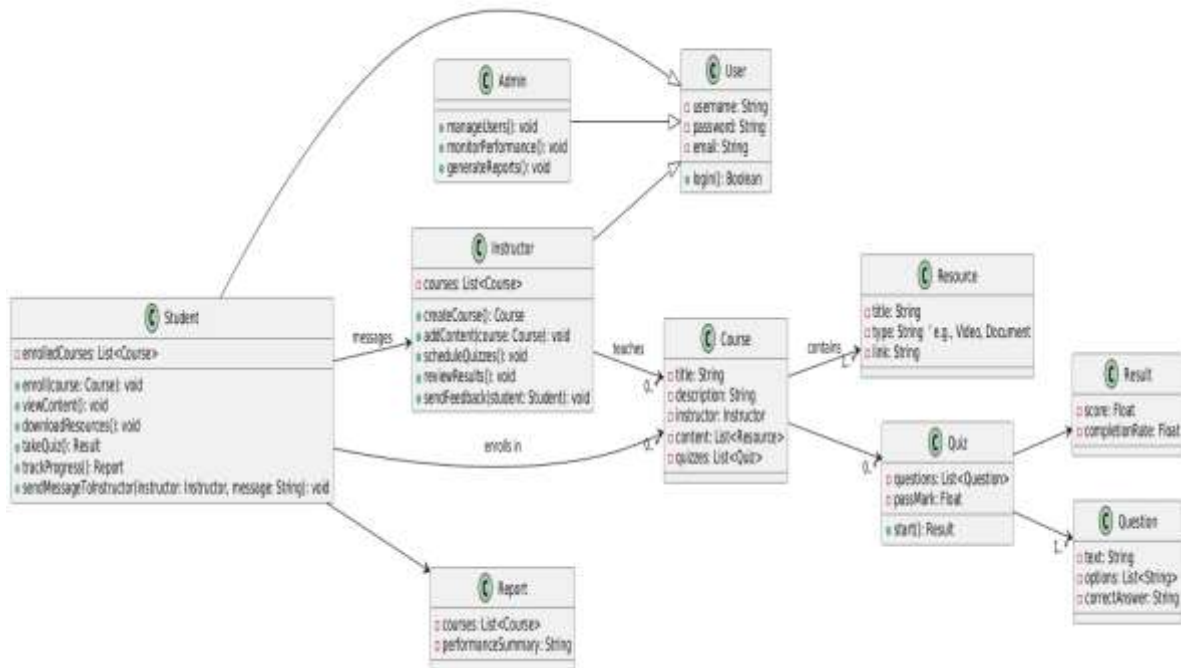
كل طالب يملك نتيجة واحدة لكل اختبار يقوم به. العلاقة تربط الطالب بالاختبار من خلال كيان النتيجة "Result".

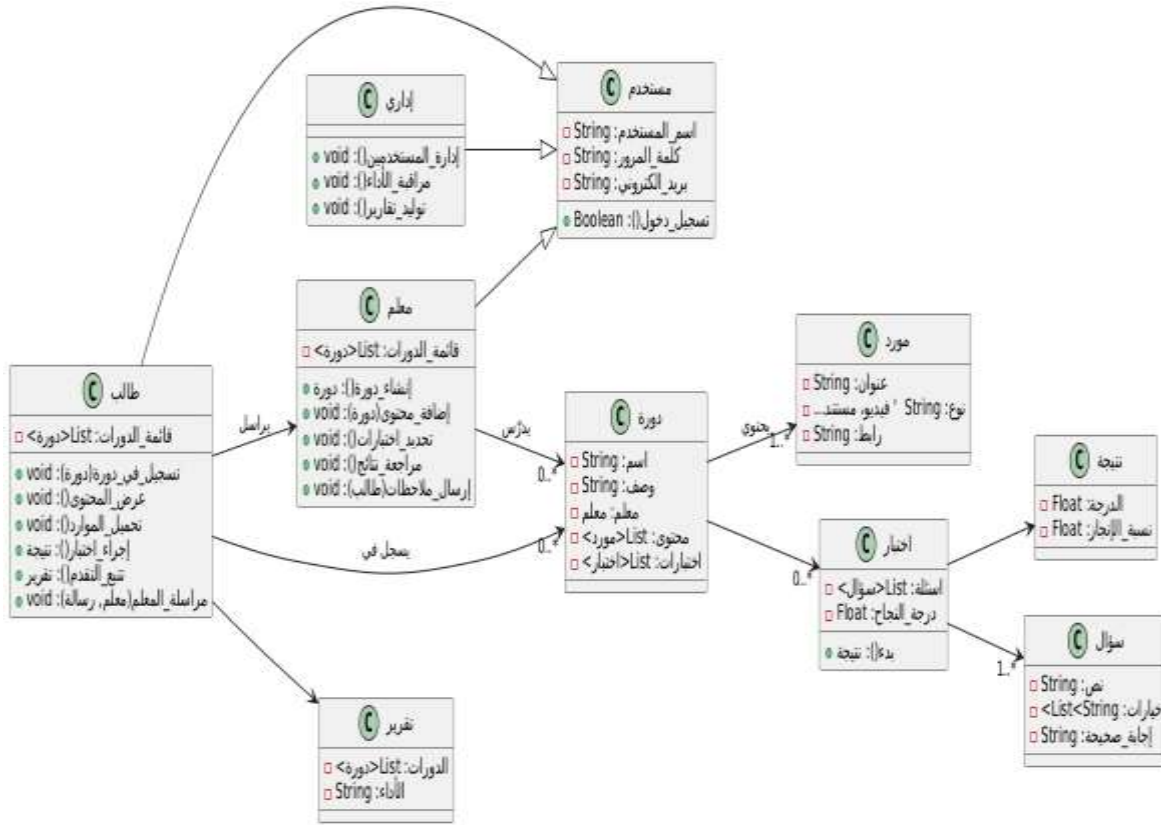
6. Student ↔ Instructor (N إلى N)

من خلال نظام المراسلة، يمكن للطالب والمعلم تبادل الرسائل الثنائية. هذه العلاقة متعددة من الطرفين.

7. Admin → Users (1 إلى N)

الإداري يمكنه إدارة عدة مستخدمين من طلاب ومعلمين (إضافة، حذف، تعديل).





شرح التوابيع الموجودة ضمن مخطط الصفوف:

الدالة	الشرح
login()	يحاول يسجل دخول المستخدم من خلال اسم المستخدم وكلمة السر. إذا كانت صحيحة بيرجع true، وإذا لا بيرجع false
enroll(course)	يسجل الطالب في دورة معينة.
viewContent()	يعرض محتوى الدورة (فيديوهات – ملفات)
downloadResources()	يتيح تحميل الموارد المرتبطة بالدورة.

takeQuiz()	يفتح اختبار نهاية الدورة، ويعيد نتيجة الاختبار (Result object).
TrackProgress()	يعرض مدى تقدم الطالب في الدورات على شكل تقرير (Report object)
sendMessageToInstructor()	يرسل رسالة للمدرّس
createCourse()	ينشئ دورة جديدة ويرجع كائن Course
addContent(course)	يضيف محتوى للدورة (مثل ملفات أو فيديوهات)
scheduleQuizzes()	يحدد مواعيد الاختبارات للطلاب
reviewResults()	يراجع نتائج اختبارات الطلاب.
sendFeedback(student)	يرسل ملاحظات أو تعليقات للطلاب
manageUsers()	يضيف/يحذف مستخدمين (طلاب، مدرّسين)
monitorPerformance()	يراقب أداء النظام أو المستخدمين.
generateReports()	ينشئ تقارير إحصائية عن التقدم أو الاستخدام
start()	يبدأ الاختبار للطلاب، ويرجع نتيجة (Result)

الان لنشكل مخطط الكيانات ERD:

أولاً: الكيانات:

User.1 (المستخدم الأساسي):

الخصائص:

username : String.1

password : String.2

email : String.3

login() : Boolean.4

Student .2

الخصائص:

enrolledCourses : List<Course> .

الوظائف:

enrollCourse(course : Course) : void .1

viewContent() : void .2

downloadResources() : void .3

takeQuiz() : Result .4

trackProgress() : .5

ReportendMessageToInstructor(instructor : Instructor,
message : String) : void

Instructor.3

الخصائص:

courses : List<Course>

الوظائف:

createCourse() : Course .1

addContent() : void .2

scheduleQuizzes() : void .3

reviewResults() : void .4

sendFeedback(student : Student) : void .5

Admin.4

الخصائص:

courses : List<Course>

الوظائف:

manageUsers() : void .1

monitorPerformance() : void .2

generateReports() : void .3

Course.5

الخصائص:

title : String .1

description : String .2

instructor : Instructor .3

content : List<Resource> .4

quizzes : List<Quiz> .5

العلاقات:

. يتبع لمعلم

. مسجل فيه طلاب

. يحتوي على محتوى واختبارات

Resource.6

الخصائص:

title : String .1

type : String .2

link : String .3

Quiz.7

الخصائص:

questions : List<Question>.1

passMark : Float.2

الوظائف:

start() : Result

Question.8

الخصائص:

text : String .1

options : List<String> .2
correctAnswer : String .3

Result.9

الخصائص:

score : Float.1
completionRate : Float.2

Report.10

الخصائص:

courses : List<Course>.1
performanceSummary : String .2

ثانياً: العلاقات:

- الطالب (Student) يمكن أن يسجل في عدة دورات (Course)
- كل دورة (Course) يمكن أن يسجل فيها عدة طلاب.
- علاقة Many-to-Many بين Student و Course

-
- المعلم (Instructor) يمكنه إنشاء أكثر من دورة (Course)
 - علاقة One-to-Many من Instructor إلى Course
-

. كل دورة (Course) تحتوي على عدة موارد تعليمية (Resource)
علاقة One-to-Many من Course إلى Resource

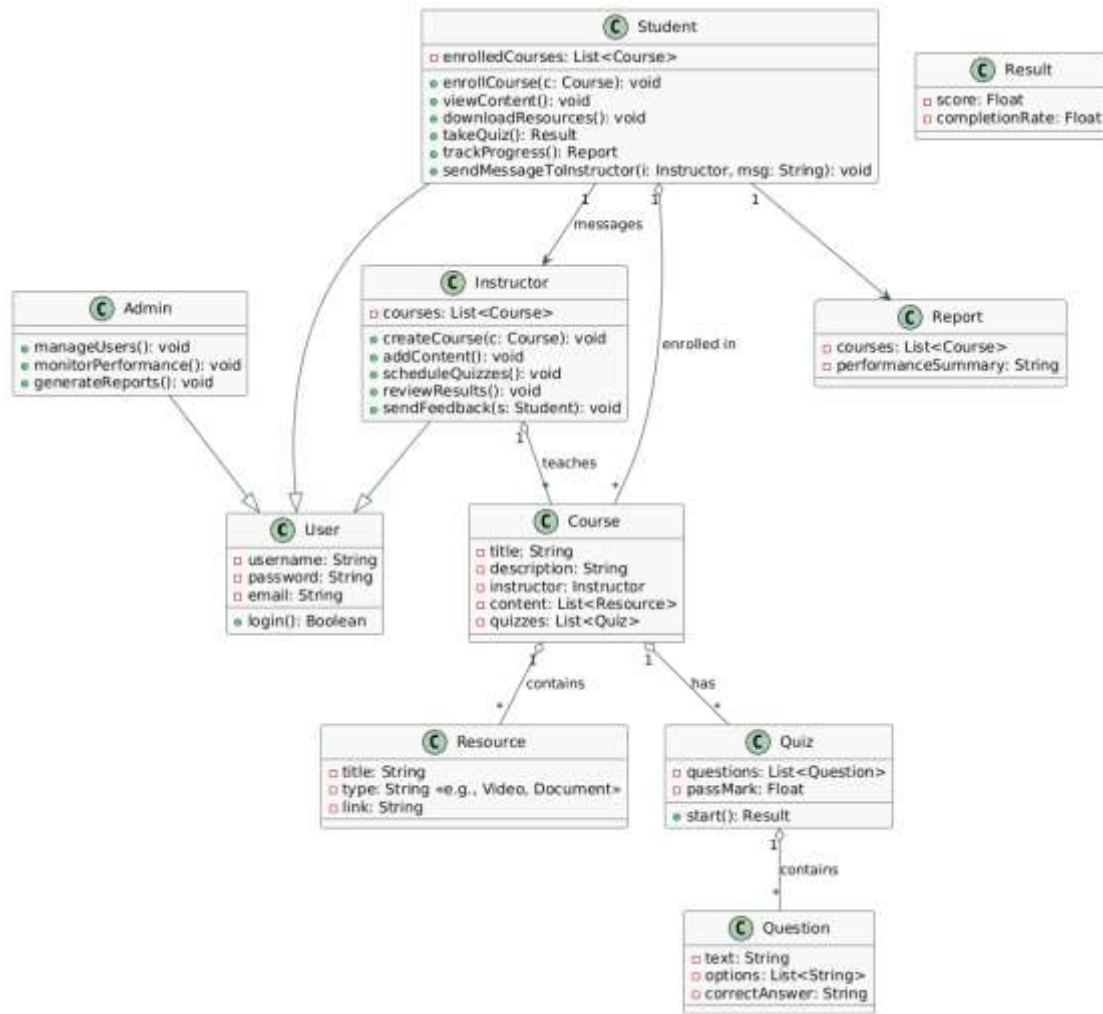
. كل دورة (Course) تحتوي على عدة اختبارات (Quiz)
علاقة One-to-Many من Course إلى Quiz

. كل اختبار (Quiz) يتكوّن من عدة أسئلة (Question)
علاقة One-to-Many من Quiz إلى Question

. كل طالب (Student) يمكن أن يأخذ اختبارات ويُنْتِج عنها نتائج (Result)
علاقة One-to-Many من Student إلى Result

. الطالب (Student) يمكن أن يحصل على تقرير أداء (Report) يُغطي جميع الدورات.
علاقة One-to-One أو One-to-Many حسب التصميم.

. الطالب (Student) يمكنه أن يرسل رسائل للمعلم (Instructor)
علاقة Many-to-One من Student إلى Instructor

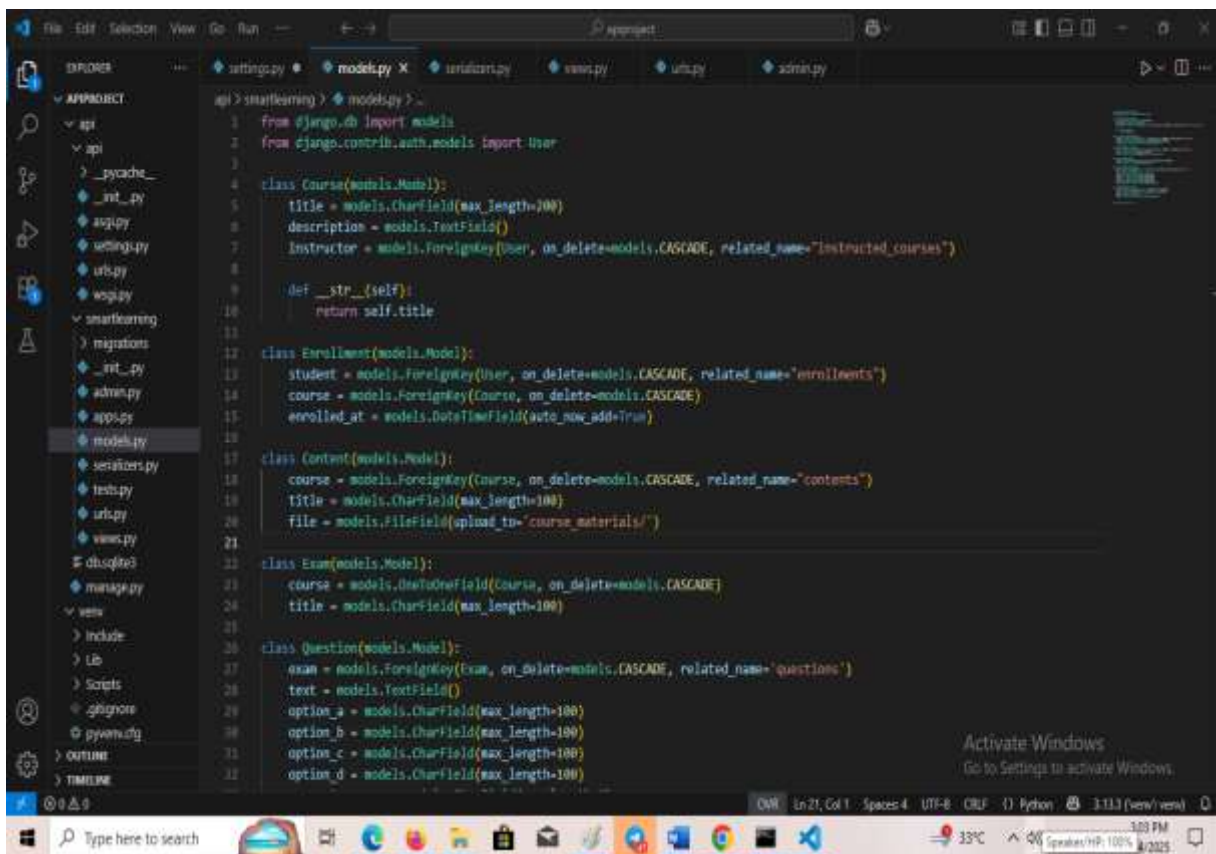


الان سنبدأ بتشكيل

API (RESTful)

قمنا بإنشاء مشروع باسم apiproject بداخله تطبيق باسم smartlearning

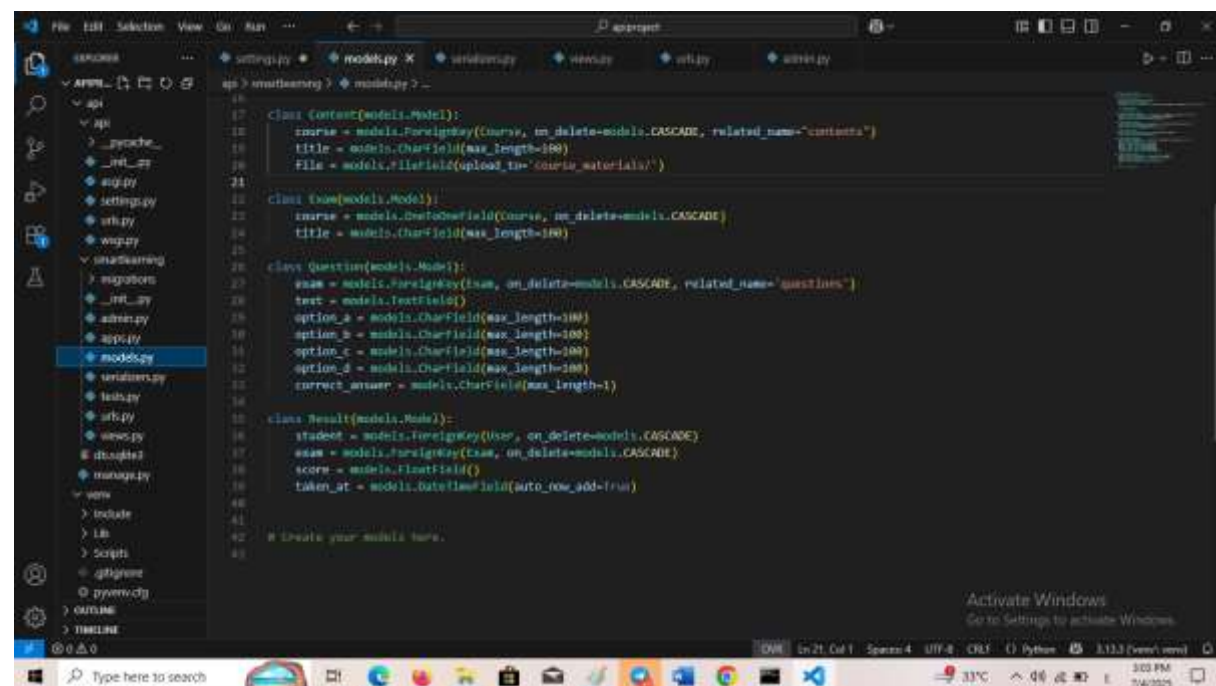
داخل ال models :



The screenshot shows a Visual Studio Code editor with a Django project named 'smartlearning'. The 'models.py' file is open, displaying the following code:

```
api > smartlearning > models.py > ...
1 from django.db import models
2 from django.contrib.auth.models import User
3
4 class Course(models.Model):
5     title = models.CharField(max_length=100)
6     description = models.TextField()
7     instructor = models.ForeignKey(User, on_delete=models.CASCADE, related_name="instructed_courses")
8
9     def __str__(self):
10         return self.title
11
12 class Enrollment(models.Model):
13     student = models.ForeignKey(User, on_delete=models.CASCADE, related_name="enrollments")
14     course = models.ForeignKey(Course, on_delete=models.CASCADE)
15     enrolled_at = models.DateTimeField(auto_now_add=True)
16
17 class Content(models.Model):
18     course = models.ForeignKey(Course, on_delete=models.CASCADE, related_name="contents")
19     title = models.CharField(max_length=100)
20     file = models.FileField(upload_to="course_materials/")
21
22 class Exam(models.Model):
23     course = models.ForeignKey(Course, on_delete=models.CASCADE)
24     title = models.CharField(max_length=100)
25
26 class Question(models.Model):
27     exam = models.ForeignKey(Exam, on_delete=models.CASCADE, related_name="questions")
28     text = models.TextField()
29     option_a = models.CharField(max_length=100)
30     option_b = models.CharField(max_length=100)
31     option_c = models.CharField(max_length=100)
32     option_d = models.CharField(max_length=100)
```

The file explorer on the left shows the project structure, including 'api', 'smartlearning', and 'migrations' folders. The status bar at the bottom indicates the file is at line 21, column 1, with 4 spaces, UTF-8 encoding, and CR/LF line endings.

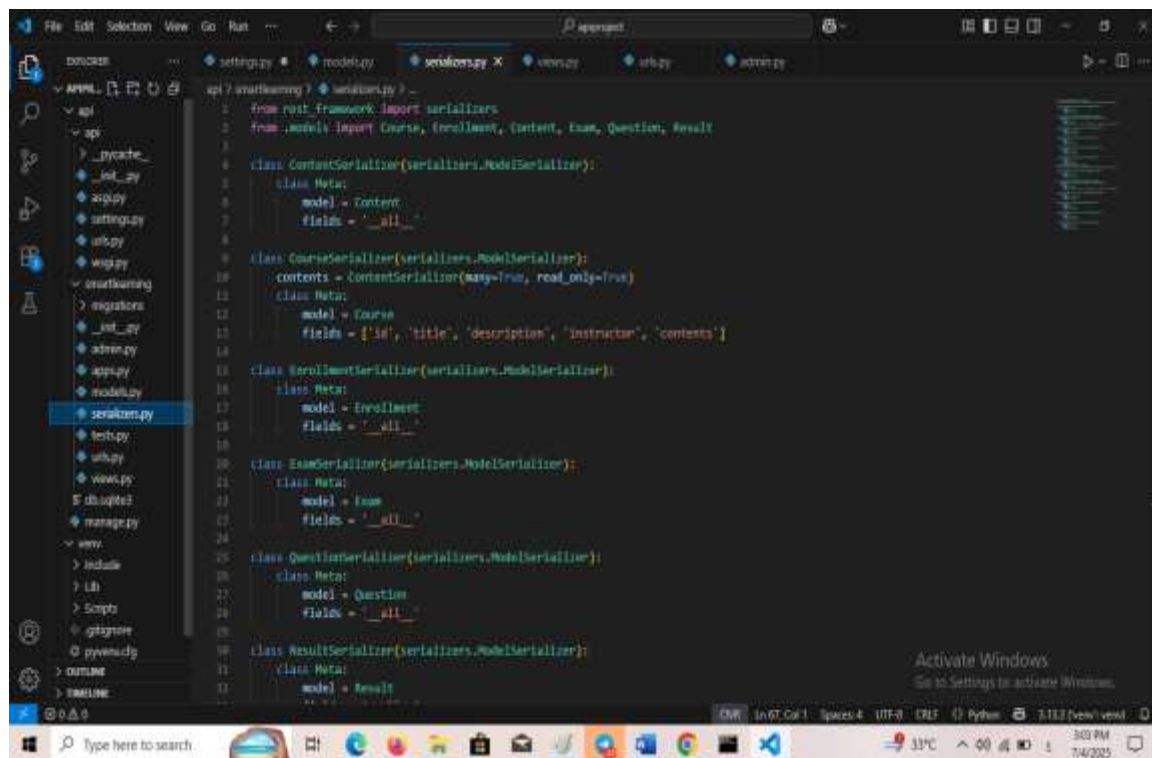


The screenshot shows the same Visual Studio Code editor with the 'models.py' file. The code is as follows:

```
api > smartlearning > models.py > ...
17
18 class Content(models.Model):
19     course = models.ForeignKey(Course, on_delete=models.CASCADE, related_name="contents")
20     title = models.CharField(max_length=100)
21     file = models.FileField(upload_to="course_materials/")
22
23 class Exam(models.Model):
24     course = models.ForeignKey(Course, on_delete=models.CASCADE)
25     title = models.CharField(max_length=100)
26
27 class Question(models.Model):
28     exam = models.ForeignKey(Exam, on_delete=models.CASCADE, related_name="questions")
29     text = models.TextField()
30     option_a = models.CharField(max_length=100)
31     option_b = models.CharField(max_length=100)
32     option_c = models.CharField(max_length=100)
33     option_d = models.CharField(max_length=100)
34     correct_answer = models.CharField(max_length=1)
35
36 class Result(models.Model):
37     student = models.ForeignKey(User, on_delete=models.CASCADE)
38     exam = models.ForeignKey(Exam, on_delete=models.CASCADE)
39     score = models.FloatField()
40     taken_at = models.DateTimeField(auto_now_add=True)
41
42 # Create your models here.
43
```

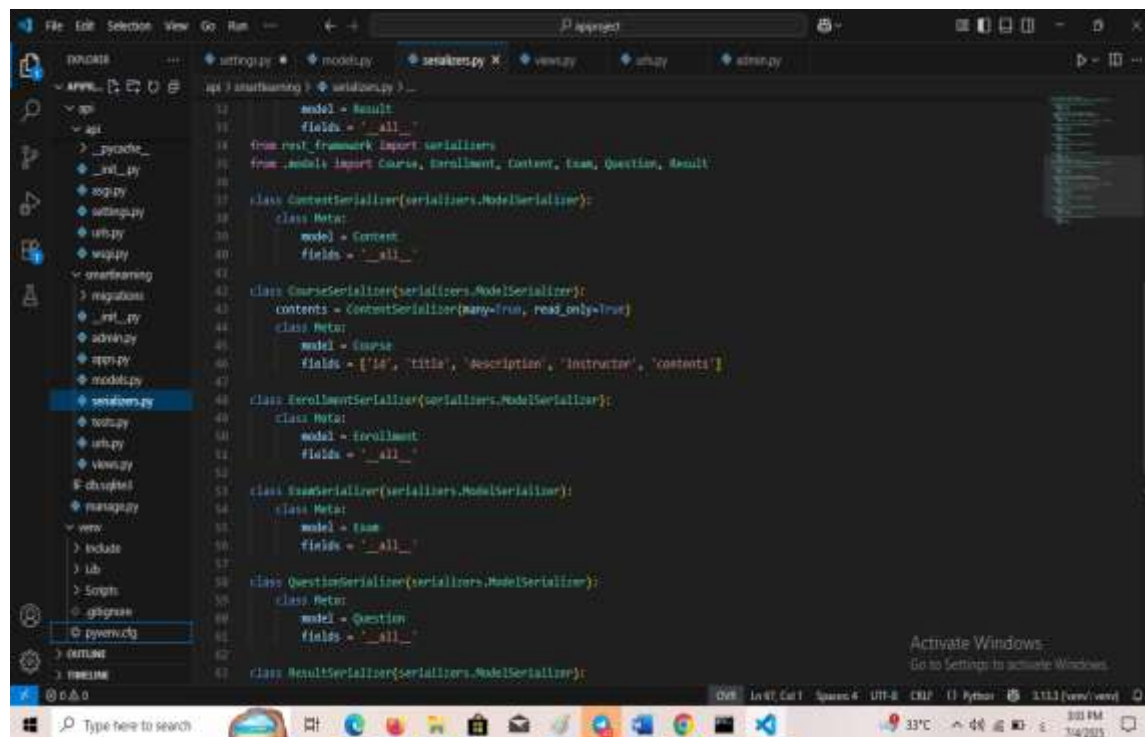
The file explorer on the left shows the project structure, including 'api', 'smartlearning', and 'migrations' folders. The status bar at the bottom indicates the file is at line 42, column 1, with 4 spaces, UTF-8 encoding, and CR/LF line endings.

: serializer.py أما داخل ال



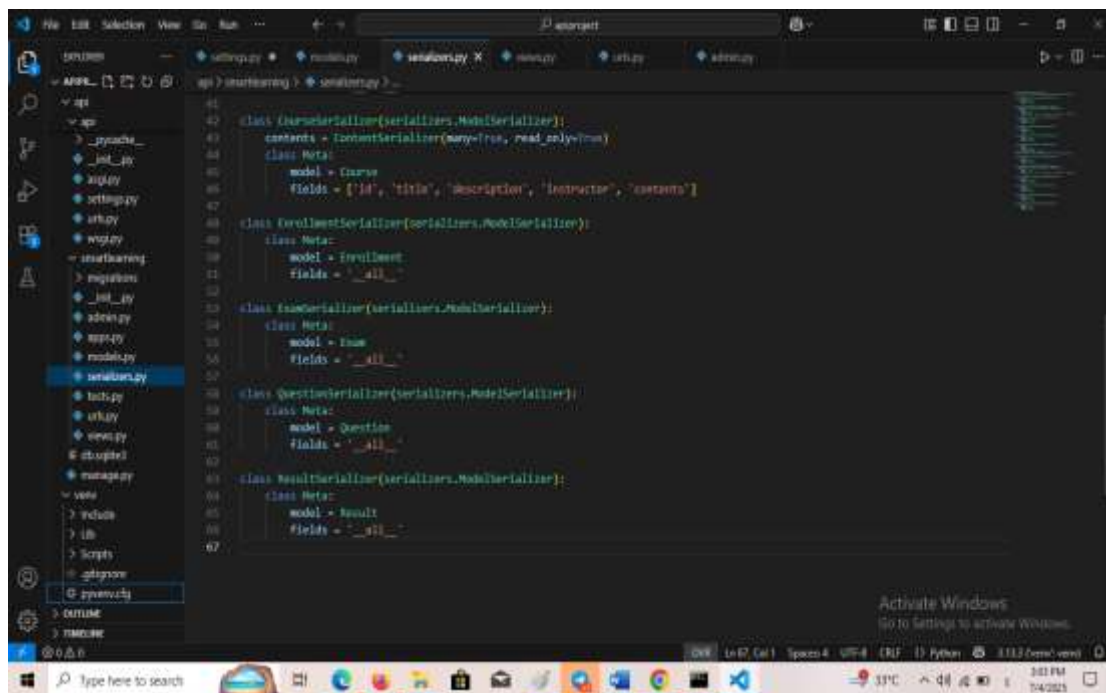
This screenshot shows the Visual Studio Code editor with the file `serializers.py` open. The file contains several Django REST Framework serializer classes. The left sidebar shows the project structure with `serializers.py` selected. The bottom status bar shows the file encoding as UTF-8 and the Python version as 3.11.2.

```
1 from rest_framework import serializers
2 from .models import Course, Enrollment, Content, Exam, Question, Result
3
4 class ContentSerializer(serializers.ModelSerializer):
5     class Meta:
6         model = Content
7         fields = '__all__'
8
9 class CourseSerializer(serializers.ModelSerializer):
10     contents = ContentSerializer(many=True, read_only=True)
11     class Meta:
12         model = Course
13         fields = ['id', 'title', 'description', 'instructor', 'contents']
14
15 class EnrollmentSerializer(serializers.ModelSerializer):
16     class Meta:
17         model = Enrollment
18         fields = '__all__'
19
20 class ExamSerializer(serializers.ModelSerializer):
21     class Meta:
22         model = Exam
23         fields = '__all__'
24
25 class QuestionSerializer(serializers.ModelSerializer):
26     class Meta:
27         model = Question
28         fields = '__all__'
29
30 class ResultSerializer(serializers.ModelSerializer):
31     class Meta:
32         model = Result
```



This screenshot shows the same Visual Studio Code editor with `serializers.py` open. In this view, the `Result` model is selected in the `fields` list of the `ResultSerializer` class. The project structure in the sidebar is the same, and the status bar also shows UTF-8 encoding and Python 3.11.2.

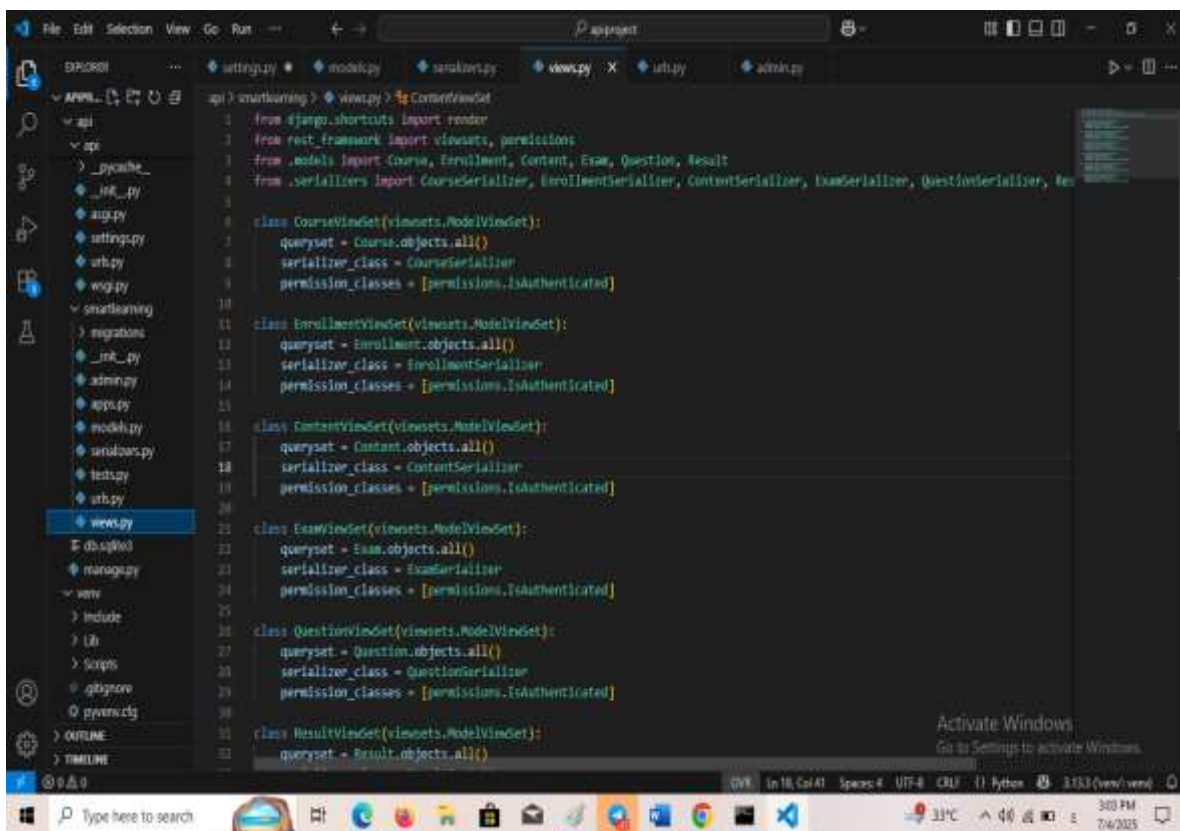
```
32 model = Result
33 fields = '__all__'
34
35 from rest_framework import serializers
36 from .models import Course, Enrollment, Content, Exam, Question, Result
37
38 class ContentSerializer(serializers.ModelSerializer):
39     class Meta:
40         model = Content
41         fields = '__all__'
42
43 class CourseSerializer(serializers.ModelSerializer):
44     contents = ContentSerializer(many=True, read_only=True)
45     class Meta:
46         model = Course
47         fields = ['id', 'title', 'description', 'instructor', 'contents']
48
49 class EnrollmentSerializer(serializers.ModelSerializer):
50     class Meta:
51         model = Enrollment
52         fields = '__all__'
53
54 class ExamSerializer(serializers.ModelSerializer):
55     class Meta:
56         model = Exam
57         fields = '__all__'
58
59 class QuestionSerializer(serializers.ModelSerializer):
60     class Meta:
61         model = Question
62         fields = '__all__'
63
64 class ResultSerializer(serializers.ModelSerializer):
65     class Meta:
```



A screenshot of the Visual Studio Code editor showing the file `serializers.py` in the `api > smartlearning >` directory. The file contains several Django REST Framework serializers. The left sidebar shows the project structure with `serializers.py` selected. The bottom status bar shows the Python version as 3.11.2 (venv/venv).

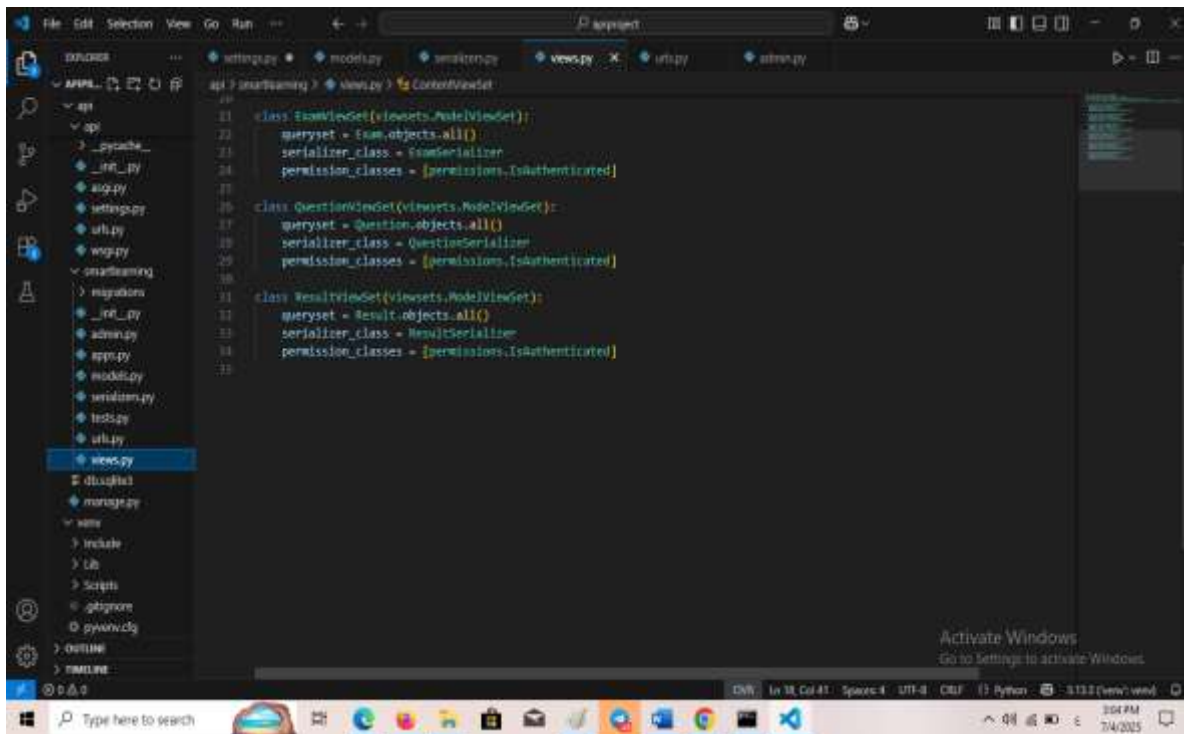
```
41 class CourseSerializer(serializers.ModelSerializer):
42     contents = ContentSerializer(many=True, read_only=True)
43     class Meta:
44         model = Course
45         fields = ('id', 'title', 'description', 'instructor', 'contents')
46
47 class EnrollmentSerializer(serializers.ModelSerializer):
48     class Meta:
49         model = Enrollment
50         fields = '__all__'
51
52 class ExamSerializer(serializers.ModelSerializer):
53     class Meta:
54         model = Exam
55         fields = '__all__'
56
57 class QuestionSerializer(serializers.ModelSerializer):
58     class Meta:
59         model = Question
60         fields = '__all__'
61
62 class ResultSerializer(serializers.ModelSerializer):
63     class Meta:
64         model = Result
65         fields = '__all__'
```

الان داخل ال views نضع:

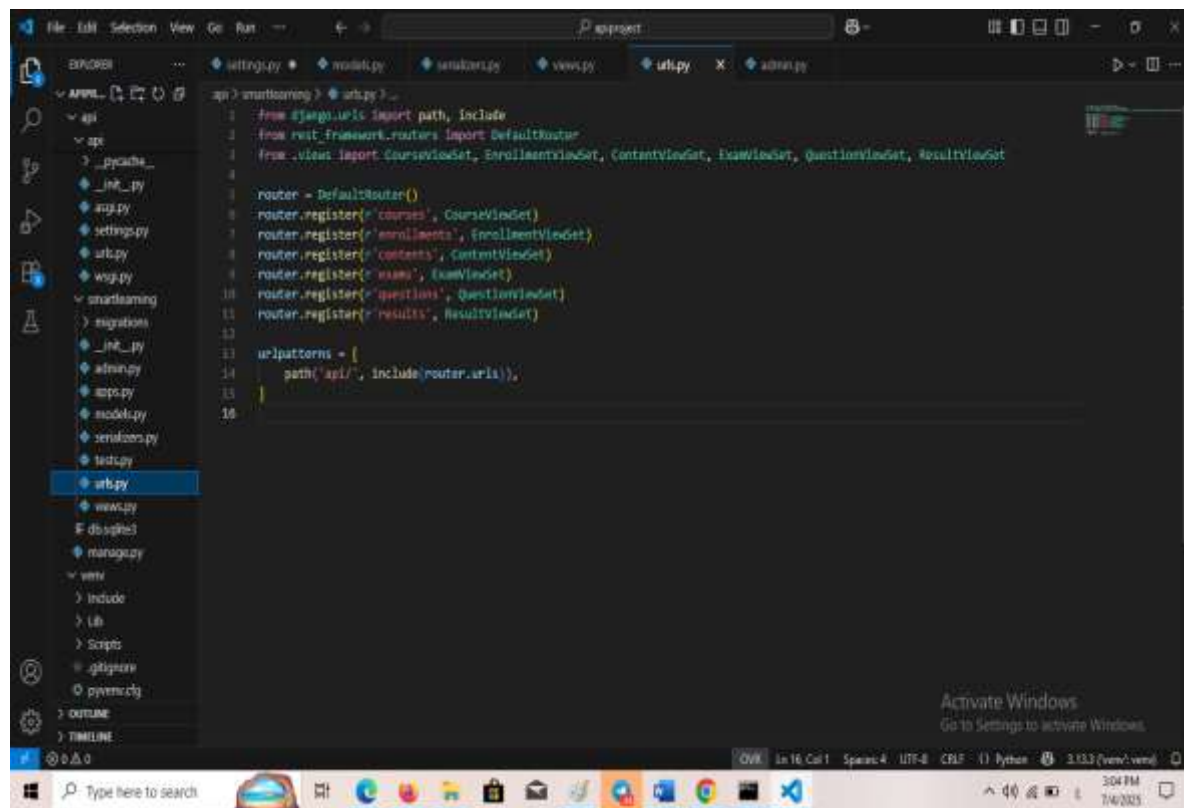


A screenshot of the Visual Studio Code editor showing the file `views.py` in the `api > smartlearning >` directory. The file contains Django REST Framework viewsets. The left sidebar shows the project structure with `views.py` selected. The bottom status bar shows the Python version as 3.11.2 (venv/venv).

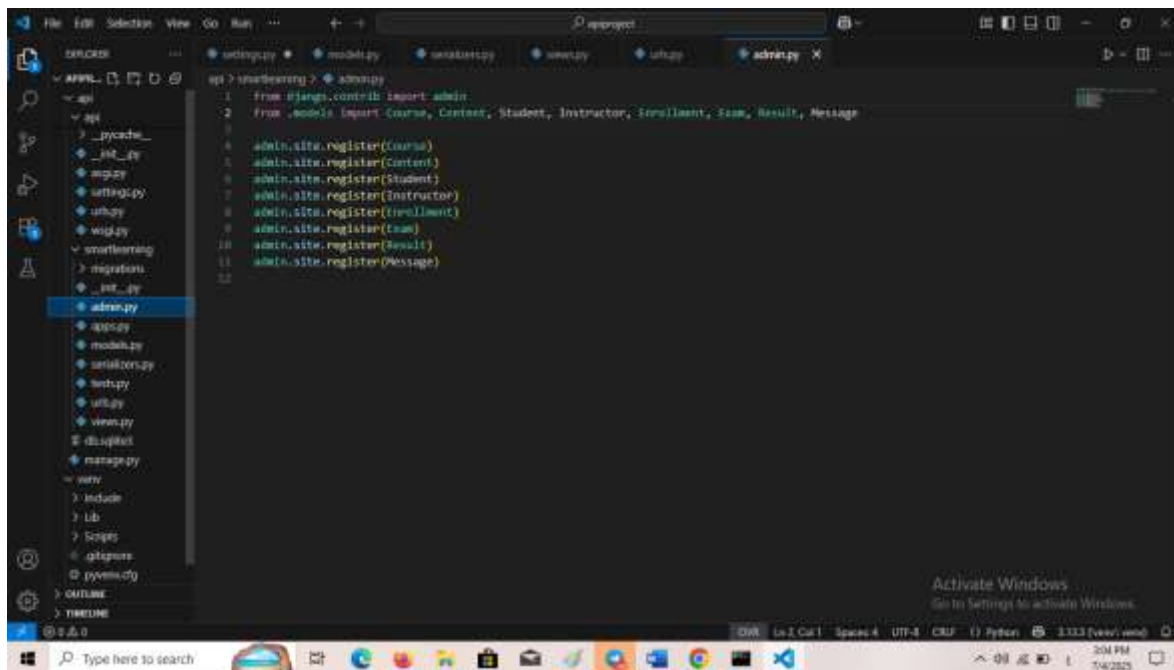
```
1 from django.shortcuts import render
2 from rest_framework import viewsets, permissions
3 from .models import Course, Enrollment, Content, Exam, Question, Result
4 from .serializers import CourseSerializer, EnrollmentSerializer, ContentSerializer, ExamSerializer, QuestionSerializer, ResultSerializer
5
6 class CourseViewSet(viewsets.ModelViewSet):
7     queryset = Course.objects.all()
8     serializer_class = CourseSerializer
9     permission_classes = [permissions.IsAuthenticated]
10
11 class EnrollmentViewSet(viewsets.ModelViewSet):
12     queryset = Enrollment.objects.all()
13     serializer_class = EnrollmentSerializer
14     permission_classes = [permissions.IsAuthenticated]
15
16 class ContentViewSet(viewsets.ModelViewSet):
17     queryset = Content.objects.all()
18     serializer_class = ContentSerializer
19     permission_classes = [permissions.IsAuthenticated]
20
21 class ExamViewSet(viewsets.ModelViewSet):
22     queryset = Exam.objects.all()
23     serializer_class = ExamSerializer
24     permission_classes = [permissions.IsAuthenticated]
25
26 class QuestionViewSet(viewsets.ModelViewSet):
27     queryset = Question.objects.all()
28     serializer_class = QuestionSerializer
29     permission_classes = [permissions.IsAuthenticated]
30
31 class ResultViewSet(viewsets.ModelViewSet):
32     queryset = Result.objects.all()
```

الان داخل ال urls.py :



وداخل ال admin.py:



```
1 from django.contrib import admin
2 from .models import Course, Content, Student, Instructor, Enrollment, Exam, Result, Message
3
4 admin.site.register(Course)
5 admin.site.register(Content)
6 admin.site.register(Student)
7 admin.site.register(Instructor)
8 admin.site.register(Enrollment)
9 admin.site.register(Exam)
10 admin.site.register(Result)
11 admin.site.register(Message)
```

هذا المثال يوضح نظام **API متكامل** لمنصة تعليم ذكية (Intelligent Learning System) مبني باستخدام **Django Rest Framework**. الهدف من هذا الـ API هو تمكين المستخدمين (طلاب، معلمين، إداريين) من التفاعل مع النظام بشكل برمجي، من خلال تطبيق ويب أو تطبيق جوال.

شرح مكونات المشروع :

models.py.1

في هذا الملف نعرف قاعدة البيانات:

- كل دورة لها معلم واحد. (instructor)
- كل طالب يمكن أن يسجل في أكثر من دورة (عن طريق جدول Enrollment).

- كل دورة تحتوي على محتوى تعليمي.
- الدورة يمكن أن تحتوي على اختبار (Exam) والأسئلة (Question) المرتبطة به.
- نتيجة الاختبار محفوظة في جدول Result

[:serializers.py.2](#)

يحدد كيف يتم تحويل البيانات بين الكائنات (Models) وJSON. يستخدم في إرسال واستقبال البيانات عبر الـAPI.

[:views.py.3](#)

يحتوي على الوظائف التي تتعامل مع الطلبات HTTP مثل:

- 1. GET
- 2. POST
- 3. PUT
- 4. DELETE

[:urls.py.4](#)

يربط الطلبات القادمة بالوظائف المناسبة داخل الـViews. يستخدم Router لعمل روابط جاهزة لكل كيان مثل:

- /api/courses/
- /api/enrollments/
- /api/results/

كيف يستخدمه المعلم؟

- يدخل دورات جديدة باستخدام /api/courses/.
- يضيف محتوى إلى الدورة باستخدام /api/contents/.

• ينشئ اختبار وأسئلة للدورة.

كيف يستخدمه المعلم؟

• يدخل دورات جديدة باستخدام `/api/courses/`.

• يضيف محتوى إلى الدورة باستخدام `/api/contents/`.

• ينشئ اختبار وأسئلة للدورة.