Final Proje: Çevrimiçi Kurs Uygulamasına Yeni Bir Değerlendirme Özelliği Ekle

Gerekli tahmini süre: 90 dakika

Yeni bir tam yığın geliştirici olarak, lider geliştiriciniz size yeni bir kurs değerlendirme özelliğini uygulama görevini verdi. Bu özelliği başarılı bir şekilde sunmak için, gerekli modelleri, şablonları ve görünümleri tasarlamak ve geliştirmek için Django tam yığın becerilerinizi kullanacaksınız. Son olarak, çevrimiçi kurs uygulamanızı çalıştıracak ve işlevselliğini sağlamak için kapsamlı bir şekilde test edeceksiniz.

Hedefler

Bu laboratuvarın sonunda şunları yapabileceksiniz:

1. Yeni kurs değerlendirme özelliğinin gereksinimlerini anlamak
2. Soru, seçenek ve gönderim modelleri oluşturmak
3. Yönetim sitesini kullanarak sınav ile ilgili modellerle yeni bir kurs nesnesi oluşturmak
4. Soruları ve seçenekleri göstermek için kurs detayları şablonunu güncellemek
5. Gönderim sonucunu göstermek için yeni bir sınav sonucu şablonu oluşturmak
6. Yeni bir sınav sonucu gönderim görünümü oluşturmak
7. Sınav sonucunu görüntülemek ve değerlendirmek için yeni bir görünüm oluşturmak

Bulut IDE'de Dosyalarla Çalışma

Bulut IDE'ye yeni iseniz, bu bölüm size Bulut IDE'deki projenizin bir parçası olan dosyaları nasıl oluşturup düzenleyeceğinizi gösterecektir.

Bulut IDE içinde dosyalarınızı ve dizinlerinizi görüntülemek için dosya simgesine tıklayarak açabilirsiniz.

Eğer `git clone` komutunu kullanarak başlangıç kodunu klonladıysanız, aşağıdaki resme benzeyecektir:

Eğer klonlamadıysanız ve boş bir projeyle başlıyorsanız, bu şekilde görünecektir:

Yeni Bir Dosya Oluşturma

Projenizde yeni bir dosya oluşturmak için sağ tıklayın ve Yeni Dosya seçeneğini seçin. Aynı işlemi `File -> New File` seçeneği ile de yapabilirsiniz.

Yeni dosyanın adını vermeniz istenecek. Bu durumda, adını `sample.html` koyalım.

Dizin yapısındaki `sample.html` dosya adına tıkladığınızda, dosya sağ panelde açılacaktır. Farklı türde dosyalar oluşturabilirsiniz; örneğin, JavaScript dosyaları için `FILE_NAME.js`.

Aşağıdaki örnekte, bazı temel HTML kodlarını yapıştırdık ve ardından dosyayı kaydettik.

Bu dosyayı şu şekilde kaydediyoruz:

- Menüden giderek
- Mac'te `Command + S` veya Windows'ta `CTRL + S` tuşlarına basarak
- Alternatif olarak, çalışmanızı otomatik olarak kaydedecektir

Kurulum: Bir Uygulama Oluştur

1. Editörün menüsünden bir terminal penceresi açın: **Terminal > Yeni Terminal** seçeneğini seçin.

2. Eğer şu anda proje klasöründe değilseniz, proje klasörünüze geçmek için aşağıdaki kodu kopyalayıp yapıştırın. Kodu kopyalamak için kodun sağındaki kopyala butonunu seçin.

   ```
   cd /home/project
   ```

3. Eğer Git deposu henüz mevcut değilse, bu projeye gerekli başlangıç kodunu içeren Git deposunu klonlamak için aşağıdaki komutu çalıştırın.

   ```
   [ ! -d 'tfjzl-final-cloud-app-with-database' ] && git clone https://github.com/ibm-developer-skills-network/tfjzl-final-cloud-app-with-data
   ```

4. Laboratuvara başlamak için **tfjzl-final-cloud-app-with-database** dizinine geçin.

```
cd tfjzl-final-cloud-app-with-database
```

5. Bu dizinin içeriğini listeleyerek laboratuvar için gereken belgeleri görün.

```
ls
```

6. Gerekli tüm paketleri içerecek sanal bir ortam oluşturalım.

```
pip install --upgrade distro-info
pip3 install --upgrade pip==23.2.1
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate
```

7. Python çalışma zamanını ayarlayın ve şablon projesini test edin.

```
pip install -U -r requirements.txt
```

8. İlk göçleri oluşturun ve veritabanı şemasını oluşturun:

Göçler, Django'nun modellerinizde yaptığınız değişiklikleri (bir alan eklemek, bir modeli silmek vb.) veritabanı şemanıza yayma yoludur. Çoğunlukla otomatik olarak tasarlanmışlardır, ancak göçlerin ne zaman yapılacağını, ne zaman çalıştırılacağını ve karşılaşabileceğiniz yaygın sorunları bilmeniz gerekecek. Göçlerle ve Django'nun veritabanı şemasını yönetimiyle etkileşimde bulunmak için kullanacağınız birkaç komut vardır:

1. ***migrate***, göçleri uygulamaktan ve geri almaktan sorumludur
2. ***makemigrations***, modellerinizde yaptığınız değişikliklere dayanarak yeni göçler oluşturmaktan sorumludur
3. ***sqlmigrate***, bir göç için SQL ifadelerini gösterir
4. ***showmigrations***, bir projenin göçlerini ve durumunu listeler

```
python3 manage.py makemigrations
python3 manage.py migrate
```

9. Bu sefer sunucuyu başarıyla çalıştırın.

```
python3 manage.py runserver
```

Uygulamayı Başlat

12. Aşağıdaki resme benzer görünecek:

13. Terminalinizde, web sunucunuzu durdurmak için `CTRL+C` tuşlarına basın.

Bir Git Repo ile Çalışmak

Laboratuvar ortamının geçici olduğunu anlamak önemlidir. Yok olmadan önce kısa bir süre boyunca var olur. Gerekli olduğunda, yeni bir laboratuvar ortamında yeniden oluşturmak için yaptığınız tüm değişiklikleri kendi GitHub deposuna göndermelisiniz.

Ayrıca, bu ortamın paylaşıldığını ve dolayısıyla güvenli olmadığını unutmayın. Bu ortamda kişisel bilgilerinizi, kullanıcı adlarınızı, şifrelerinizi veya erişim jetonlarınızı herhangi bir amaçla saklamamalısınız.

Tekrar iş yapmaktan korunmak için, kodunuzu zaman zaman bir GitHub deposuna göndermeli ve taahhüt etmelisiniz.

Değişiklikleri gözden geçirme

Yapılan değişiklikleri gözden geçirmek için terminalde aşağıdaki komutları çalıştırın:

```
cd [your repo name]
git status
```

Değişiklikleri taahhüt et

Artık yaptığınız değişiklikleri taahhüt etmeniz gerekiyor. Bunu yapmadan önce, yeni ve revize edilmiş dosyaları taahhüde eklemeniz gerekiyor:

```
git add sample.html
git add existing_file.html
```

Dosyaları ekledikten sonra `git status` komutunu yeniden çalıştırın.

Git kurulumu - Kimliğiniz

Git kullanmaya başladığınızda yapmanız gereken ilk şey, kullanıcı adınızı ve e-posta adresinizi ayarlamaktır. Bu önemlidir çünkü her Git taahhüdü bu bilgiyi kullanır ve oluşturduğunuz taahhütlerin bir parçası olacaktır. Verilen Kullanıcı Adı ve E-posta Kimliği'ni kişisel bilgilerinize göre değiştirin:

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Değişiklikleri Kaydet

Artık yaptığınız değişiklikleri kaydedebilirsiniz. Değişiklikleri kaydetmek için aşağıdaki komutu çalıştırın. `-m` seçeneğini kullanarak bir commit mesajı geçeceksiniz.

```
git commit -m 'changes made from the lab environment'
```

Git Uzak

1. GitHub üzerinde ücretsiz bir hesap oluşturun.
2. E-posta adresinizi doğrulayın, eğer daha önce yapmadıysanız.
3. Ayarlar'a gidin: Herhangi bir sayfanın sağ üst köşesinde profil fotoğrafınıza tıklayın, ardından Ayarlar'a tıklayın.
4. Geliştirici Ayarları'na gidin.
5. Ortamınızdan GitHub'a kimlik doğrulamak için kişisel erişim belirteci oluşturun.
6. Belirteç için ayrıcalıkları ayarlayın ve oluşturun.
7. Kodunuzu itmek için bir uzak depo oluşturun.
8. Bir depoyu `git clone` ile kopyaladığınızda, otomatik olarak kopyalanan depoya geri işaret eden origin adlı bir uzak bağlantı oluşturur. Bu, merkezi bir deponun yerel bir kopyasını oluşturan geliştiriciler için yararlıdır çünkü yukarı akıştaki değişiklikleri çekmek veya yerel taahhütleri yayınlamak için kolay bir yol sağlar.

   Eğer uzak origin zaten ayarlandıysa (bu, GitHub'dan kopyaladığınızda en olası durumdur).
   Aşağıdaki komutlardaki `YOUR_GITHUB_USER` kısmını gerçek GitHub kullanıcı adınızla değiştirin.

   ```
   git remote set-url origin https://github.com/YOUR_GITHUB_USER/my-course-repo.git
   ```

   Uzak origin'i ilk kez oluşturmak (yerelde boş bir depo ile başladığınızda).

   ```
   git remote add origin https://github.com/YOUR_GITHUB_USER/my-course-repo.git
   ```

9. Bu depoya kod eklemek için birden fazla seçenekle karşılaşacaksınız. Laboratuvar ortamlarınızda, size şablon kod sağlanacaktır, bu nedenle en iyi seçenek **komut satırından mevcut bir depoyu itmek** olacaktır.

   Aşağıdaki komutları çalıştırırken, bir `kullanıcı adı` girmeniz istenecektir. Bu, GitHub kullanıcı adınız olacaktır. Ve `şifre`, daha önce oluşturduğunuz erişim belirteci olacaktır.

   Daha sonra:

   ```
   git branch -M main
   ```

```
git push -u origin main
```

Görev 1: Yeni Modeller Oluştur

`onlinecourse/models.py` dosyasında birkaç yeni model oluşturmanız gerekecek.

Soru modeli

`Question` modeli, aşağıdaki özelliklere sahip bir sınavın sorularını kaydedecektir:

- Bir kurs için soruları kalıcı hale getirmek için kullanılır
- Kurs ile Çoktan-Bire ilişkisi vardır
- Soru metni içerir
- Her soru için bir not puanı vardır

▼ İpucu

```
class Question(models.Model):
    Kurs için yabancı anahtar
    Soru metni
    Soru notu
```

▼ Çözüm

```
class Question(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    content = models.CharField(max_length=200)
    grade = models.IntegerField(default=50)
    def __str__(self):
        return "Soru: " + self.content
```

Not Al

Ayrıca, puanı hesaplayan aşağıdaki fonksiyonu Soru modelinize ekleyebilirsiniz:

```
# method to calculate if the learner gets the score of the question
def is_get_score(self, selected_ids):
    all_answers = self.choice_set.filter(is_correct=True).count()
    selected_correct = self.choice_set.filter(is_correct=True, id__in=selected_ids).count()
    if all_answers == selected_correct:
        return True
    else:
        return False
```

Seçim modeli

Bir `Choice` modeli, bir sorunun tüm seçimlerini kaydeder:

- `Question` modeli ile Çoktan-Bire ilişki
- Seçim metni
- Bu seçimin doğru olup olmadığını belirtir

▼ İpucu

```
class Choice(models.Model):
    Soruya yabancı anahtar
    Seçim içeriği metin olarak
    Seçim doğru mu boolean olarak
```

▼ Çözüm

```
class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    content = models.CharField(max_length=200)
    is_correct = models.BooleanField(default=False)
```

Gönderim Modeli

Yorum satırlarıyla kapatılmış `Submission` modeline sahipsiniz, bu model şunları içerir:

- Sınav Gönderimleri ile Çoktan-Bire ilişkiler, örneğin, birden fazla sınav gönderimi bir kurs kaydına ait olabilir.
- Seçimler veya sorular ile Çoktan-Çok ilişkisi. Basitlik açısından, gönderimi Choice modeli ile ilişkilendirebilirsiniz.

Seçilen seçimleri ilişkilendirmek için `Submission` modelinin yorumunu kaldırmanız ve kullanmanız gerekiyor.

`models.py` dosyasındaki diğer modellere örnek olarak bakabilirsiniz.

Referansınız için bir ER Diyagramı örneği:

Nihai çözüm

Ayrıca, aşağıda sağlanan nihai çözümü inceleyebilirsiniz.

▼ Son onlinecourse/models.py dosyasını görmek için buraya tıklayın

```
import sys
from django.utils.timezone import now
try:
    from django.db import models
except Exception:
    print("There was an error loading django modules. Do you have django installed?")
    sys.exit()
from django.conf import settings
import uuid
# Instructor model
class Instructor(models.Model):
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
    )
    full_time = models.BooleanField(default=True)
    total_learners = models.IntegerField()
    def __str__(self):
        return self.user.username
# Learner model
class Learner(models.Model):
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
    )
    STUDENT = 'student'
    DEVELOPER = 'developer'
    DATA_SCIENTIST = 'data_scientist'
    DATABASE_ADMIN = 'dba'
    OCCUPATION_CHOICES = [
        (STUDENT, 'Student'),
        (DEVELOPER, 'Developer'),
        (DATA_SCIENTIST, 'Data Scientist'),
        (DATABASE_ADMIN, 'Database Admin')
    ]
    occupation = models.CharField(
        null=False,
        max_length=20,
        choices=OCCUPATION_CHOICES,
        default=STUDENT
    )
    social_link = models.URLField(max_length=200)
    def __str__(self):
```

```
        return self.user.username + "," + \
               self.occupation
# Course model
class Course(models.Model):
    name = models.CharField(null=False, max_length=30, default='online course')
    image = models.ImageField(upload_to='course_images/')
    description = models.CharField(max_length=1000)
    pub_date = models.DateField(null=True)
    instructors = models.ManyToManyField(Instructor)
    users = models.ManyToManyField(settings.AUTH_USER_MODEL, through='Enrollment')
    total_enrollment = models.IntegerField(default=0)
    is_enrolled = False
    def __str__(self):
        return "Name: " + self.name + "," + \
               "Description: " + self.description
# Lesson model
class Lesson(models.Model):
    title = models.CharField(max_length=200, default="title")
    order = models.IntegerField(default=0)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    content = models.TextField()
# Enrollment model
# <HINT> Once a user enrolled a class, an enrollment entry should be created between the user and course
# And we could use the enrollment to track information such as exam submissions
class Enrollment(models.Model):
    AUDIT = 'audit'
    HONOR = 'honor'
    BETA = 'BETA'
    COURSE_MODES = [
        (AUDIT, 'Audit'),
        (HONOR, 'Honor'),
        (BETA, 'BETA')
    ]
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    date_enrolled = models.DateField(default=now)
    mode = models.CharField(max_length=5, choices=COURSE_MODES, default=AUDIT)
    rating = models.FloatField(default=5.0)
class Question(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    content = models.CharField(max_length=200)
    grade = models.IntegerField(default=50)
    def __str__(self):
        return "Question: " + self.content
    def is_get_score(self, selected_ids):
        all_answers = self.choice_set.filter(is_correct=True).count()
        selected_correct = self.choice_set.filter(is_correct=True, id__in=selected_ids).count()
        if all_answers == selected_correct:
            return True
        else:
            return False
class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    content = models.CharField(max_length=200)
    is_correct = models.BooleanField(default=False)
class Submission(models.Model):
    enrollment = models.ForeignKey(Enrollment, on_delete=models.CASCADE)
    choices = models.ManyToManyField(Choice)
```

Göçleri çalıştırın

```
python3 manage.py makemigrations onlinecourse
python3 manage.py migrate
```

Not: Model göçleriyle ilgili herhangi bir hata görürseniz, mevcut veritabanı `db.sqlite3`'ü silebilir ve yukarıdaki göçü tekrar çalıştırabilirsiniz.

Değerlendirme: Aşağıdaki Soru, Seçenek ve Gönderim modelinin bir ekran görüntüsünü alın ve ekran görüntüsünü `01-models.png` olarak kaydedin.

Görev 2: Model Değişikliklerini Kaydet

Artık oluşturduğunuz yeni özellikleri kullanabilmek için `onlinecourse/admin.py` dosyasında değişiklikler yapacaksınız.

Open **admin.py** in IDE

Yeni modelleri içe aktar

Şu anda `onlinecourse/admin.py` dosyasında yalnızca `Course`, `Lesson`, `Instructor` ve `Learner` modellerini içe aktarıyorsunuz.

`Question`, `Choice` ve `Submission` eklemeniz gerekiyor.

▼ Çözüm

```
from .models import Course, Lesson, Instructor, Learner, Question, Choice, Submission
```

QuestionInline ve ChoiceInline oluştur

Admin sitesinde bunları birlikte düzenleyebilmeniz için `QuestionInline` ve `ChoiceInline` sınıflarını oluşturun.

▼ İpucu

```
class Class_Name(admin.StackedInline):
    model = Model_Name
    extra = 2
```

▼ Çözüm

```
class ChoiceInline(admin.StackedInline):
    model = Choice
    extra = 2
class QuestionInline(admin.StackedInline):
    model = Question
    extra = 2
```

QuestionAdmin sınıfını oluştur

▼ İpucu

```
class QuestionAdmin(admin.ModelAdmin):
    inlines = [Question_sub_content]
    list_display = ['content']
```

▼ Çözüm

```
class QuestionAdmin(admin.ModelAdmin):
    inlines = [ChoiceInline]
    list_display = ['content']
```

Question, Choice ve Submission'ı kaydet

Yeni modelleri kaydettikten sonra, admin sitesi aracılığıyla dersler, sorular ve soru seçenekleri ile yeni bir kurs oluşturabilirsiniz.

Kayıt dekoratörü: `register(*models, site=django.contrib.admin.sites.site)`

▼ İpucu

```
admin.site.register(Model1, Model2)
admin.site.register(Model3)
```

▼ Çözüm

```
admin.site.register(Question, QuestionAdmin)
admin.site.register(Choice)
admin.site.register(Submission)
```

Son `admin.py` dosyasını burada görebilirsiniz:

▼ Çözüm

```
from django.contrib import admin
# <İP UCU> Buraya yeni Modelleri içe aktarın
from .models import Course, Lesson, Instructor, Learner, Question, Choice, Submission
# <İP UCU> Buraya QuestionInline ve ChoiceInline sınıflarını kaydedin
class LessonInline(admin.StackedInline):
    model = Lesson
    extra = 5
class ChoiceInline(admin.StackedInline):
    model = Choice
    extra = 2
class QuestionInline(admin.StackedInline):
    model = Question
    extra = 2
# Modellerinizi buraya kaydedin.
class CourseAdmin(admin.ModelAdmin):
    inlines = [LessonInline]
    list_display = ('name', 'pub_date')
    list_filter = ['pub_date']
    search_fields = ['name', 'description']
class QuestionAdmin(admin.ModelAdmin):
    inlines = [ChoiceInline]
    list_display = ['content']
class LessonAdmin(admin.ModelAdmin):
    list_display = ['title']
# <İP UCU> Buraya Question ve Choice modellerini kaydedin
admin.site.register(Course, CourseAdmin)
admin.site.register(Lesson, LessonAdmin)
admin.site.register(Instructor)
admin.site.register(Learner)
admin.site.register(Question, QuestionAdmin)
admin.site.register(Choice)
admin.site.register(Submission)
```

Değerlendirme: `admin.py` dosyasının ekran görüntüsünü alın ve ekran görüntüsünü `02-admin-file.png` olarak kaydedin.

Bir admin kullanıcısı oluştur

Aşağıdaki bilgilerle bir admin kullanıcısı oluşturalım:

1. Kullanıcı adı: `admin`
2. E-posta adresi: *boş bırakmak için enter'a basın*
3. Şifre: Seçiminiz, ya da `p@ssword123` kullanın

```
python3 manage.py createsuperuser
```

Değişikliklerinizi Kaydedin

Django geliştirme sunucusunu çalıştırın ve admin sitesini kullanarak Soru ve Seçenek nesnelerini ekleyip ekleyemediğinizi kontrol edin.

```
python3 manage.py runserver
```

Django yönetim panelini başlat

Değerlendirme: Yönetim sitesinin ekran görüntüsünü alın ve ekran görüntüsünü `03-admin-site.png` olarak kaydedin.

Görev 3: Kurs Detay Şablonunu Güncelle

Artık sınav bölümü oluşturmak için kurs detay şablonunu güncelleyeceksiniz; bu bölümde bir soru ve seçenekler listesi olacak.

Bir sınav birden fazla soru içerir ve her birinin birden fazla doğru cevabı olmalıdır (çoktan seçmeli).

Değişiklikler `templates/onlinecourse/course_details_bootstrap.html` dosyasında yapılacaktır.

Open **course_detail_bootstrap.html** in IDE

Aşağıda verilen yer tutucuda kodu düzenlemeye başlayın:

1. Kullanıcı kimlik doğrulaması yapılmışsa, kurs sınavını bir soru ve seçenekler listesi ile gösterin:

   ▼ İpucu

   ```
   {% if CONDITION %}
   </br>
   <!-- Kalan kod burada olacak -->
   {% endif %}
   ```

   ▼ Çözüm

   ```
   {% if user.is_authenticated %}
   </br>
   <!-- Kalan kod burada olacak -->
   {% endif %}
   ```

2. Sınavı başlatmak için bir buton ekleyin:

   ▼ İpucu

   ```
   <TAG class="CLASS CLASS-primary CLASS-block" data-toggle="collapse" data-target="#exam">Sınavı Başlat</TAG>
   ```

▼ Çözüm

```
<button class="btn btn-primary btn-block" data-toggle="collapse" data-target="#exam">Sınavı Başlat</button>
```

3. Katlanabilir bir `div` ekleyin:

▼ İpucu

```
<TAG id="exam" class="collapse">
</TAG>
```

▼ Çözüm

```
<div id="exam" class="collapse">
</div>
```

4. Soru mantığını bir `form` içine ekleyin:

▼ İpucu

```
<div id="exam" class="collapse">
    <form id="questionform" action="{% url 'onlinecourse:submit' course.id %}" method="POST">
        KURS SORULARINI BURADA DÖNGÜYE AL
    </form>
</div>
```

▼ Çözüm

```
<div id="exam" class="collapse">
    <form id="questionform" action="{% url 'onlinecourse:submit' course.id %}" method="POST">
        {% for question in course.question_set.all %}
            <!-- Soru UI bileşenleri burada olacak -->
        {% endfor %}
    </form>
</div>
```

5. Soru UI'sini ekleyin:

▼ İpucu

```
<div class="card mt-1">
    <div class="card-header"><h5>{{ question.PROPERTY }}</h5></div>
    {% csrf_token %}
    <div class="form-group">
    </div>
</div>
```

▼ Çözüm

```
<div class="card mt-1">
    <div class="card-header"><h5>{{ question.content }}</h5></div>
    {% csrf_token %}
    <div class="form-group">
        <!-- Seçenek bileşenleri burada olacak -->
    </div>
</div>
```

6. Seçenek bileşenlerini ekleyin:

▼ İpucu

```
{% for ITEM in question.FIELD.all %}
<div class="form-check">
    <label class="form-check-label">
        <input type="checkbox" name="choice_{{choice.IDENTIFIER}}"
               class="form-check-input" id="{{choice.IDENTIFIER}}"
               value="{{choice.IDENTIFIER}}">{{ choice.FIELD }}
    </label>
</div>
{% endfor %}
```

▼ Çözüm

```
{% for choice in question.choice_set.all %}
<div class="form-check">
    <label class="form-check-label">
        <input type="checkbox" name="choice_{{choice.id}}"
               class="form-check-input" id="{{choice.id}}"
               value="{{choice.id}}">{{ choice.content }}
    </label>
</div>
{% endfor %}
```

Nihai çözüm

Nihai çözümü burada görüntüleyin:

▼ Çözüm

```
{% if user.is_authenticated %}
</br>
<button class="btn btn-primary btn-block" data-toggle="collapse" data-target="#exam">Sınavı Başlat</button>
<div id="exam" class="collapse">
    <form id="questionform" action="{% url 'onlinecourse:submit' course.id %}" method="POST">
        {% for question in course.question_set.all %}
        <div class="card mt-1">
            <div class="card-header">
                <h5>{{ question.content }}</h5>
            </div>
            {% csrf_token %}
            <div class="form-group">
                {% for choice in question.choice_set.all %}
                <div class="form-check">
                    <label class="form-check-label">
                        <input type="checkbox" name="choice_{{choice.id}}" class="form-check-input"
                            id="{{choice.id}}" value="{{choice.id}}">{{ choice.content }}
                    </label>
                </div>
                {% endfor %}
            </div>
        </div>
        {% endfor %}
        <input class="btn btn-success btn-block" type="submit" value="Gönder">
    </form>
</div>
```

```
    {% endif %}
```

Test etmek için çalıştırın:

```
python3 manage.py runserver
```

Çevrimiçi kurs uygulamasını başlat

Bu anda, sınavı gönderemezsiniz. Bunu bir sonraki laboratuvar çalışmasında uygulayacaksınız.

Kodunuzu Taahhüt Edin

Kodunuzu GitHub'a taahhüt etmek ve göndermek, kaybetmemek için iyi bir uygulamadır.

Değerlendirme: `course_details_bootstrap.html` dosyasının ekran görüntüsünü alın ve ekran görüntüsünü `04-course-details.png` olarak kaydedin.

Görev 4: Test Verisi

Artık uygulamanız için test verisi oluşturacaksınız.

Eğitmen ekle

`admin` olarak bir Eğitmen ekleyin

Kurs bilgileri

Alan	**Değer**
İsim	Django Öğrenimi
Resim	buradan indirin
Açıklama	Django, Python ile yazılmış son derece popüler ve tam özellikli bir sunucu tarafı web çerçevesidir.
Yayın tarihi	Bugün
Eğitmenler	admin
Ders #1 Başlığı	Django Nedir
Ders #1 Sırası	0
Ders #1 İçeriği	Django, hızlı geliştirmeyi ve temiz, pragmatik tasarımı teşvik eden yüksek seviyeli bir Python web çerçevesidir. Deneyimli geliştiriciler tarafından oluşturulmuştur; bu nedenle, uygulamanızı yazmaya odaklanabilirsiniz, web geliştirme ile ilgili birçok zorluğu üstlenir, tekerleği yeniden icat etmenize gerek kalmaz.

Test sorusu

Alan	**Değer**
Kurs	İsim: Django Öğrenimi, Açıklama: …
İçerik	Django bir Python çerçevesi mi?
Not	100
Seçenek #1 İçeriği	Evet
Seçenek #1 Doğru mu	

Alan	Değer
Seçenek #2 İçeriği	Hayır
Seçenek #2 Doğru mu	*Boş bırakın*

Kursun ön yüzünü açalım.

Uygulamayı Başlat

Görev 5: Gönderim Değerlendirmesi

Yeni modeller oluşturduğunuz için, şimdi bunları `views.py` dosyasının en üstüne import etmeniz gerekiyor.

Open **views.py** in IDE

```
from .models import Course, Enrollment, Question, Choice, Submission
```

Gönderim görünümü

Artık form gönderimi için işlev tabanlı bir görünüm oluşturacaksınız.

Bir sınav gönderim kaydı oluşturmak için `def submit(request, course_id):` şeklinde bir gönderim görünümü oluşturun. Aşağıdaki mantığa dayanarak bunu uygulayabilirsiniz:

- Mevcut kullanıcıyı ve kurs nesnesini alın, ardından ilişkili kayıt nesnesini edinin
  (İPUCU: `Enrollment.objects.get(user=..., course=...)`)
- Kaydı referans alan yeni bir gönderim nesnesi oluşturun
  (İPUCU: `Submission.objects.create(enrollment=...)`)
- HTTP istek nesnesinden seçilen seçenekleri toplayın (İPUCU: yük payload sözlüğünü almak için `request.POST` kullanabilir ve sözlük değerlerinden seçim kimliğini alabilirsiniz. Örnek bir kod parçası da sağlanmıştır.)
- Her bir seçilen seçim nesnesini gönderim nesnesine ekleyin
- Sınav sonucunu göstermek için gönderim kimliği ile birlikte `show_exam_result` görünümüne yönlendirin
- Yeni `submit` görünümünü yönlendirmek için `urls.py` dosyasını yapılandırın, örneğin `path('<int:course_id>/submit/', ...),`

Form gönderimi `views.py` içinde

▼ İpucu

```
def submit(PARAM1, PARAM2):
    course = get_object_or_404(MODEL, pk=PARAM2)
    user = request.OBJECT
    enrollment = Enrollment.objects.get(ARUGMENT1, ARUGMENT2)
    submission = Submission.objects.create(ARGUMENT)
    choices = extract_answers(ARGUMENT)
    submission.choices.set(ARGUMENT)
    submission_id = submission.id
    return HttpResponseRedirect(reverse(viewname='onlinecourse:exam_result', args=(course_id, submission_id,)))
```

▼ Çözüm

```
def submit(request, course_id):
    course = get_object_or_404(Course, pk=course_id)
    user = request.user
    enrollment = Enrollment.objects.get(user=user, course=course)
    submission = Submission.objects.create(enrollment=enrollment)
    choices = extract_answers(request)
    submission.choices.set(choices)
    submission_id = submission.id
    return HttpResponseRedirect(reverse(viewname='onlinecourse:exam_result', args=(course_id, submission_id,)))
```

Gönderim görünümü düğmesini `urls.py` içinde yönlendirin

Open **urls.py** in IDE

▼ İpucu

```
path('<int:course_id>/FUNCTION/', views.FUNCTION, name="submit"),
```

▼ Çözüm

```
path('<int:course_id>/submit/', views.submit, name="submit"),
```

Değerlendirme görünümü

Öğrenicinin sınavı geçip geçmediğini ve soru sonuçlarını kontrol etmek için `def show_exam_result(request, course_id, submission_id):` şeklinde bir sınav sonucu görünümü oluşturun.

Görünümü aşağıdaki mantığa dayanarak uygulayabilirsiniz:

- Görünüm argümanlarındaki kimliklerine göre kurs nesnesini ve gönderim nesnesini alın
- Gönderim kaydından seçilen seçim kimliklerini alın
- Her bir seçilen seçim için doğru cevap olup olmadığını kontrol edin
- Kurs içindeki tüm soruların notlarını toplayarak toplam puanı hesaplayın
- HTML sayfasını render etmek için bağlamda kurs, seçim ve notu ekleyin
- Yeni `show_exam_result` görünümünü yönlendirmek için `urls.py` dosyasını yapılandırın, örneğin `path('course/<int:course_id>/submission/<int:submission_id>/result/', ...),`

Sınav sonuçları `views.py` içinde

▼ İpucu

```
def show_exam_result(request, PARAM1, PARAM2):
    context = {}
    course = get_object_or_404(MODEL1, pk=PARAM1)
    submission = MODEL2.objects.get(id=PARAM2)
    choices = submission.choices.all()
    total_score = 0
    questions = course.RELATION_SET.all()  # Kursun ilişkili soruları olduğunu varsayıyoruz
    for question in questions:
        correct_choices = question.RELATION_SET.filter(ARGUMENT1=True)  # Sorunun tüm doğru seçimlerini al
        selected_choices = choices.filter(ARGUMENT2=question)  # Kullanıcının soruya seçtiği seçimler
        # Seçilen seçimlerin doğru seçimlerle aynı olup olmadığını kontrol et
        if set(ARGUMENT3) == set(ARGUMENT4):
            total_score += question.ATTRIBUTE  # Tüm doğru cevaplar seçildiğinde sorunun notunu ekle
    context['KEY1'] = course
    context['KEY2'] = total_score
    context['KEY3'] = choices
    return render(request, 'TEMPLATE_PATH', context)
```

▼ Çözüm

```
def show_exam_result(request, course_id, submission_id):
    context = {}
    course = get_object_or_404(Course, pk=course_id)
    submission = Submission.objects.get(id=submission_id)
```

```
    choices = submission.choices.all()
    total_score = 0
    questions = course.question_set.all()  # Kursun ilişkili soruları olduğunu varsayıyoruz
    for question in questions:
        correct_choices = question.choice_set.filter(is_correct=True)  # Sorunun tüm doğru seçimlerini al
        selected_choices = choices.filter(question=question)  # Kullanıcının soruya seçtiği seçimler
        # Seçilen seçimlerin doğru seçimlerle aynı olup olmadığını kontrol et
        if set(correct_choices) == set(selected_choices):
            total_score += question.grade  # Tüm doğru cevaplar seçildiğinde sorunun notunu ekle
    context['course'] = course
    context['grade'] = total_score
    context['choices'] = choices
    return render(request, 'onlinecourse/exam_result_bootstrap.html', context)
```

Sınav sonuçları `urls.py` içinde

Open **urls.py** in IDE

▼ İpucu

```
path('course/<int:course_id>/submission/<int:submission_id>/result/', views.FUNCTION, name="exam_result"),
```

▼ Çözüm

```
path('course/<int:course_id>/submission/<int:submission_id>/result/', views.show_exam_result, name="exam_result"),
```

Değerlendirme: `views.py` dosyasının ekran görüntüsünü alın ve ekran görüntüsünü `05-views.png` olarak kaydedin.

Değerlendirme: `urls.py` dosyasının ekran görüntüsünü alın ve ekran görüntüsünü `06-urls.png` olarak kaydedin.

Görev 6: Sınav Sonuç Şablonunu Tamamlayarak Sınav Gönderim Sonuçlarını Göster

`exam_result_bootstrap.html` HTML şablonunu, bir öğrencinin sınavı geçip geçmediğini, toplam puan gibi detayları, her soru için sonucu ve benzeri bilgileri gösterecek şekilde tamamlayın. Daha fazla detay için önceki UI tasarımına göz atın.

Güncellenmiş şablonu, tasarım ekibinin UI tasarımına uygun hale getirmek için `Bootstrap` ile stilize edin.

Geçme çıktısı

Sınavı geçen öğrencilere, son puan ve tebrik mesajı gösterilmelidir.

▼ İpucu

```
<b>Tebrikler, {{ USER_DETAILS }}!</b> Sınavı geçtiniz ve {{ GRADE }}/100 puan ile kursu tamamladınız.
```

▼ Çözüm

```
<b>Tebrikler, {{ user.first_name }}!</b> Sınavı geçtiniz ve {{ grade }}/100 puan ile kursu tamamladınız.
```

Kalma çıktısı

Sınavı geçemeyen öğrencilere, yanlış seçimlerle birlikte son puan gösterilmelidir. Öğrencinin sınavı tekrar yapmasına ve yeniden göndermesine izin verilmelidir.

▼ İpucu

```
<b>Başarısız</b> Üzgünüm, {{ USER_DETAILS }}! Sınavı {{ GRADE }}/100 puan ile geçemediniz.
```

▼ Çözüm

```
<b>Başarısız</b> Üzgünüm, {{ user.first_name }}! Sınavı {{ grade }}/100 puan ile geçemediniz.
```

Sınav sonucu

Ayrıca, öğrencinin nasıl bir performans gösterdiğini görebilmesi için sınav sonuçlarını da göstermeniz gerekir.

▼ İpucu

```
SORULARDA DÖNGÜ
<div class="card mt-1">
    <div class="card-header"><h5>SORU DETAYLARI</h5></div>
    <div class="form-group">
        SEÇENEKLERDE DÖNGÜ
        <div class="form-check">
            EĞER SEÇENEK DOĞRU VE SEÇİLMİŞSE
            <div class="text-success">Doğru cevap: SEÇENEK DETAYLARI</div>
            Aksi takdirde EĞER SEÇENEK DOĞRU VE SEÇİLMEMİŞSE
            <div class="text-warning">Seçilmedi: SEÇENEK DETAYLARI</div>
            Aksi takdirde EĞER SEÇENEK DOĞRU DEĞİL VE SEÇİLMİŞSE
            <div class="text-danger">Yanlış cevap: SEÇENEK DETAYLARI</div>
            Aksi takdirde
            <div>SEÇENEK DETAYLARI</div>
            TÜM IF'leri SONLANDIR
        </div>
        İÇ DÖNGÜYÜ SONLANDIR
    </div>
</div>
SON DÖNGÜ
```

▼ Çözüm

```
    {% for question in course.question_set.all %}
    <div class="card mt-1">
        <div class="card-header"><h5>{{ question.content }}</h5></div>
        <div class="form-group">
            {% for choice in question.choice_set.all %}
            <div class="form-check">
                {% if choice.is_correct and choice in choices %}
                <div class="text-success">Correct answer: {{ choice.content }}</div>
                {% else %}{% if choice.is_correct and not choice in choices %}
                <div class="text-warning">Not selected: {{ choice.content }}</div>
                {% else %}{% if not choice.is_correct and choice in choices %}
                <div class="text-danger">Wrong answer: {{ choice.content }}</div>
                {% else %}
                <div>{{ choice.content }}</div>
                {% endif %}{% endif %}{% endif %}
```

```
            </div>
            {% endfor %}
        </div>
    </div>
    {% endfor %}
```

Değerlendirme: Deneme sınavından bir sonucu ekran görüntüsü alın ve ekran görüntüsünü `07-final.png` olarak kaydedin.

Özet

Bu laboratuvarda, kurs değerlendirme özelliği için bir kod şablonunu elde etme ve test etme gereksinimlerini anladınız. Bu özellikleri etkili bir şekilde uygulayarak, gereksinimlerin uyumunu sağlarken iyi bir proje organizasyonu ve belgelenmesi sürdürdünüz.

Author(s)

- Yan Luo
- Muhammad Yahya