

SOURCE CODE:

```
Settings  admin.py x
Virtual-Classroom-master > Virtual-Classroom-master > lecture > admin.py
1  from django.contrib import admin
2  from .models import CoursePack, Podcast, Video, Pdf
3  # Register your models here.
4
5  admin.site.register(CoursePack)
6  admin.site.register(Podcast)
7  admin.site.register(Video)
8  admin.site.register(Pdf)
```

```
Settings  apps.py x
Virtual-Classroom-master > Virtual-Classroom-master > lecture > apps.py > ...
1  from django.apps import AppConfig
2
3
4  class LectureConfig(AppConfig):
5  |     name = 'lecture'
6
```

```

1 from django import forms
2 from django.contrib.auth.models import User
3
4 from .models import CoursePack, Podcast, Video, Pdf, Evaluation
5
6
7 class CoursePackForm(forms.ModelForm):
8
9     class Meta:
10         model = CoursePack
11         fields = ['instructor', 'course_title', 'course_code', 'thumbnail']
12
13
14
15 class PodcastForm(forms.ModelForm):
16
17     class Meta:
18         model = Podcast
19         fields = ['material_title', 'material_file']
20
21
22 class UserForm(forms.ModelForm):
23     password = forms.CharField(widget=forms.PasswordInput)
24
25     class Meta:
26         model = User
27         fields = ['username', 'email', 'password']
28
29
30 class EvaluationForm(forms.Form):
31     post = forms.CharField()
32
33

```

```

1 from django.contrib.auth.models import Permission, User
2 from django.db import models
3
4 # Create your models here.
5
6 class CoursePack(models.Model):
7     user = models.ForeignKey(User, default=1, on_delete=models.CASCADE)
8     instructor = models.CharField(max_length=250)
9     course_title = models.CharField(max_length=500)
10    course_code = models.CharField(max_length=100)
11    thumbnail = models.FileField()
12    is_favorite = models.BooleanField(default=False)
13
14    def __str__(self):
15        return self.course_title + ' - ' + self.instructor
16
17 class Podcast(models.Model):
18     course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
19     material_title = models.CharField(max_length=300)
20     material_file = models.FileField(default='')
21     is_favorite = models.BooleanField(default=False)
22
23     def __str__(self):
24         return self.material_title
25
26
27 class Video(models.Model):
28     course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
29     video_title = models.CharField(max_length=300)
30     video_file = models.FileField(default='')
31     is_favorite = models.BooleanField(default=False)
32
33     def __str__(self):
34         return self.video_title
35

```

```

18     course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
19     material_title = models.CharField(max_length=300)
20     material_file = models.FileField(default='')
21     is_favorite = models.BooleanField(default=False)
22
23     def __str__(self):
24         return self.material_title
25
26
27 class Video(models.Model):
28     course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
29     video_title = models.CharField(max_length=300)
30     video_file = models.FileField(default='')
31     is_favorite = models.BooleanField(default=False)
32
33     def __str__(self):
34         return self.video_title
35
36
37 class Pdf(models.Model):
38     course = models.ForeignKey(CoursePack, on_delete=models.CASCADE)
39     pdf_title = models.CharField(max_length=300)
40     pdf_file = models.FileField(default='')
41     is_favorite = models.BooleanField(default=False)
42
43     def __str__(self):
44         return self.pdf_title
45
46
47
48 class Evaluation(models.Model):
49     answer = models.CharField(max_length=500)
50
51     def __str__(self):
52         return self.score
53

```

```

1 from django.urls import re_path as url
2 from . import views
3
4 app_name = 'lecture'
5
6 urlpatterns = [
7     # /lecture/
8     url(r'^$', views.index, name='index'),
9
10    # /lecture/id
11
12    # /lecture/id/favorite
13    # url(r'^(?P<course_id>[0-9]+)/favorite/$', views.favorite, name='favorite'),
14    url(r'^classroom/$', views.classroom, name='classroom'),
15    url(r'^video/$', views.video, name='video'),
16    url(r'^desktop/$', views.desktop, name='desktop'),
17    url(r'^collaboration/$', views.collaboration, name='collaboration'),
18    url(r'^evaluation/$', views.evaluation, name='evaluation'),
19    url(r'^answer/$', views.answer, name='answer'),
20    url(r'^profile/$', views.profile, name='profile'),
21    url(r'^register/$', views.register, name='register'),
22    url(r'^login_user/$', views.login_user, name='login_user'),
23    url(r'^logout_user/$', views.logout_user, name='logout_user'),
24
25
26    url(r'^(?P<course_id>[0-9]+)/$', views.detail, name='detail'),
27    url(r'^(?P<course_id>[0-9]+)/delete_podcast/(?P<podcast_id>[0-9]+)/$', views.delete_podcast, name='delete_podcast'),
28    url(r'^podcasts/(?P<filter_by>[a-zA_Z]+)/$', views.podcasts, name='podcasts'),
29
30
31
32    url(r'^create_coursepack/$', views.create_coursepack, name='create_coursepack'),
33    url(r'^(?P<course_id>[0-9]+)/create_podcast/$', views.create_podcast, name='create_podcast'),
34    url(r'^(?P<course_id>[0-9]+)/delete_podcast/(?P<podcast_id>[0-9]+)/$', views.delete_podcast, name='delete_podcast'),
35    url(r'^(?P<course_id>[0-9]+)/favorite_course/$', views.favorite_course, name='favorite course'),
36    url(r'^(?P<course_id>[0-9]+)/delete_course/$', views.delete_course, name='delete_course'),
37

```

```
Settings views.py 2 X
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop
1 import cv2
2 import numpy as np
3
4 from django.contrib.auth import authenticate, login
5 from django.contrib.auth import logout
6 from django.shortcuts import render
7 from django.http import JsonResponse
8 from django.shortcuts import render, get_object_or_404
9 from django.db.models import Q
10 from .forms import CoursePackForm, PodcastForm, UserForm
11 from .models import CoursePack, Podcast, Video, Pdf, Evaluation
12 from lecture.forms import EvaluationForm
13
14
15
16 AUDIO_FILE_TYPES = ['wav', 'mp3', 'ogg', 'mp4', 'pdf']
17 IMAGE_FILE_TYPES = ['png', 'jpg', 'jpeg', 'gif']
18
19
20 def create_coursepack(request):
21     if not request.user.is_authenticated:
22         return render(request, 'lecture/login.html')
23     else:
24         form = CoursePackForm(request.POST or None, request.FILES or None)
25         if form.is_valid():
26             course = form.save(commit=False)
27             course.user = request.user
28             course.thumbnail = request.FILES['thumbnail']
29             file_type = course.thumbnail.url.split('.')[-1]
30             file_type = file_type.lower()
31             if file_type not in IMAGE_FILE_TYPES:
32                 context = {
33                     'course': course,
34                     'form': form,
35                     'error_message': 'Image file must be PNG, JPG, or JPEG',
36                 }
37             return render(request, 'lecture/create_coursepack.html', context)
38
39         course.save()
40         return render(request, 'lecture/detail.html', {'course': course})
41
42     context = {
43         "form": form,
44     }
45     return render(request, 'lecture/create_coursepack.html', context)
46
47
48 def delete_course(request, course_id):
49     course = CoursePack.objects.get(pk=course_id)
50     course.delete()
51     courses = CoursePack.objects.filter(user=request.user)
52     return render(request, 'lecture/index.html', {'courses': courses})
53
54
55 def create_podcast(request, course_id):
56     form = PodcastForm(request.POST or None, request.FILES or None)
57     course = get_object_or_404(CoursePack, pk=course_id)
58     if form.is_valid():
59         courses_podcasts = course.podcast_set.all()
60         for p in courses_podcasts:
61             if p.material_title == form.cleaned_data.get("material_title"):
62                 context = {
63                     'course': course,
64                     'form': form,
65                     'error_message': 'You already added that podcast',
66                 }
67                 return render(request, 'lecture/create_podcast.html', context)
68         podcast = form.save(commit=False)
69         podcast.course = course
70         podcast.material_file = request.FILES['material_file']
71         file_type = podcast.material_file.url.split('.')[-1]
72         file_type = file_type.lower()
73         if file_type not in AUDIO_FILE_TYPES:
```

```
Settings views.py 2 X
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop
72     if file_type not in AUDIO_FILE_TYPES:
73         context = {
74             'course': course,
75             'form': form,
76             'error_message': 'Podcast file must be MP4, MP3, or OGG',
77         }
78         return render(request, 'lecture/create_podcast.html', context)
79
80     podcast.save()
81     return render(request, 'lecture/detail.html', {'course': course})
82 context = {
83     'course': course,
84     'form': form,
85 }
86 return render(request, 'lecture/create_podcast.html', context)
87
88
89 def podcasts(request, filter_by):
90     if not request.user.is_authenticated:
91         return render(request, 'lecture/login.html')
92     else:
93         try:
94             podcast_ids = []
95             for course in CoursePack.objects.filter(user=request.user):
96                 for podcast in course.podcast_set.all():
97                     podcast_ids.append(podcast.pk)
98             users_podcasts = Podcast.objects.filter(pk__in=podcast_ids)
99             if filter_by == 'favorites':
100                 users_podcasts = users_podcasts.filter(is_favorite=True)
101         except CoursePack.DoesNotExist:
102             users_podcasts = []
103         return render(request, 'lecture/podcasts.html', {
104             'podcast_list': users_podcasts,
105             'filter_by': filter_by,
106         })
107
108
109 def delete_podcast(request, course_id, podcast_id):
110     course = get_object_or_404(CoursePack, pk=course_id)
111     podcast = Podcast.objects.get(pk=podcast_id)
112     podcast.delete()
113     return render(request, 'lecture/detail.html', {'course': course})
114
115
116
117 def favorite(request, podcast_id):
118     podcast = get_object_or_404(Podcast, pk=podcast_id)
119     try:
120         if podcast.is_favorite:
121             podcast.is_favorite = False
122         else:
123             podcast.is_favorite = True
124         podcast.save()
125     except (KeyError, Podcast.DoesNotExist):
126         return JsonResponse({'success': False})
127     else:
128         return JsonResponse({'success': True})
129
130
131 def favorite_course(request, course_id):
132     course = get_object_or_404(CoursePack, pk=course_id)
133     try:
134         if course.is_favorite:
135             course.is_favorite = False
136         else:
137             course.is_favorite = True
138         course.save()
139     except (KeyError, CoursePack.DoesNotExist):
140         return JsonResponse({'success': False})
141     else:
142         return JsonResponse({'success': True})
143
```

```
Settings views.py 2 X
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop
144
145 def index(request):
146     if not request.user.is_authenticated:
147         return render(request, 'lecture/login.html')
148     else:
149         courses = CoursePack.objects.filter(user=request.user)
150         podcast_results = Podcast.objects.all()
151         query = request.GET.get("q")
152         if query:
153             courses = courses.filter(
154                 Q(course_title__icontains=query) |
155                 Q(instructor__icontains=query)
156             ).distinct()
157             podcast_results = podcast_results.filter(
158                 Q(material_title__icontains=query)
159             ).distinct()
160             return render(request, 'lecture/index.html', {
161                 'courses': courses,
162                 'podcasts': podcast_results,
163             })
164         else:
165             return render(request, 'lecture/index.html', {'courses': courses})
166
167
168
169
170
171 def detail(request, course_id):
172     if not request.user.is_authenticated:
173         return render(request, 'lecture/login.html')
174     else:
175         user = request.user
176         course = get_object_or_404(CoursePack, pk=course_id)
177         return render(request, 'lecture/detail.html', {'course': course, 'user': user})
178
179
180
181
182 def classroom(request):
183     return render(request, 'lecture/classroom.html')
184
185
186 def video(request):
187     cap = cv2.VideoCapture(0)
188     fourcc = cv2.VideoWriter_fourcc(*'XVID')
189     out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640,480))
190
191     while True:
192         ret, frame = cap.read()
193         out.write(frame) #saving
194         cv2.imshow('frame',frame)
195
196         if cv2.waitKey(1) & 0xFF == ord('q'):
197             break
198
199     cap.release()
200     out.release() #saving
201     cv2.destroyAllWindows()
202     return render(request, 'lecture/video.html')
203
204
205 def desktop(request):
206     import cv2
207     import numpy as np
208     from PIL import ImageGrab
209
210     fourcc = cv2.VideoWriter_fourcc(*'XVID')
211     out = cv2.VideoWriter("screen.avi", fourcc, 5.0, (1366, 760))
212
213     while True:
214         img = ImageGrab.grab()
215         img_np = np.array(img)
216         cv2.imshow("Screen", img_np)
217         out.write(img_np)
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



```
Settings views.py 2 X
Virtual-Classroom-master > Virtual-Classroom-master > lecture > views.py > desktop
216     cv2.imshow("Screen", img_np)
217     out.write(img_np)
218
219     if cv2.waitKey(1) == 27:
220         break
221
222     out.release()
223     cv2.destroyAllWindows()
224     return render(request, 'lecture/desktop.html')
225
226
227
228
229 def collaboration(request):
230     return render(request, 'lecture/collaboration.html')
231
232
233
234
235 def evaluation(request):
236     return render(request, 'lecture/evaluation.html')
237
238 def answer(request):
239     print("Form submitted")
240     answer = request.POST["answer_area"]
241
242     evaluation = Evaluation(answer=answer)
243     evaluation.save()
244     return render(request, 'lecture/evaluation.html')
245
246 def profile(request):
247     return render(request, 'lecture/profile.html')
248
249
250 def logout_user(request):
251     logout(request)
252     form = UserForm(request.POST or None)
253
254
255
256
257
258
259 def login_user(request):
260     if request.method == "POST":
261         username = request.POST['username']
262         password = request.POST['password']
263         user = authenticate(username=username, password=password)
264         if user is not None:
265             if user.is_active:
266                 login(request, user)
267                 courses = CoursePack.objects.filter(user=request.user)
268                 return render(request, 'lecture/index.html', {'courses': courses})
269             else:
270                 return render(request, 'lecture/login.html', {'error_message': 'Your account has been disabled'})
271         else:
272             return render(request, 'lecture/login.html', {'error_message': 'Invalid login'})
273     return render(request, 'lecture/login.html')
274
275
276 def register(request):
277     form = UserForm(request.POST or None)
278     if form.is_valid():
279         user = form.save(commit=False)
280         username = form.cleaned_data['username']
281         password = form.cleaned_data['password']
282         user.set_password(password)
283         user.save()
284         user = authenticate(username=username, password=password)
285         if user is not None:
```