Important: Beta Release coming soon!

Source code for api.models.ticket

University of Nebraska Tutoring Portal

CSLC PORTAL BACKEND

- About
- API Endpoints
- Database Models
- API Scripts
- API Config

CSLC PORTAL FRONTEND

- About

```
from django.db import models
from django.db.models.query import QuerySet




class TicketManager(models.Manager):


    def get_all(self) -> QuerySet:
        return super().get_queryset()





    def get_student(self, student: str) -> QuerySet:
        return super().get_queryset().filter(student=student)





    def get_professor(self, professor: str) -> QuerySet:
        return super().get_queryset().filter(professor=professor)





    def get_tutor(self, tutor: str) -> QuerySet:
        return super().get_queryset().filter(tutor=tutor)





    def get_section(self, section: str) -> QuerySet:
        return super().get_queryset().filter(section=section)





    def get_course(self, course: str) -> QuerySet:
        return super().get_queryset().filter(section__course=course)
```

```
    def get_active(self) -> QuerySet:
        return super().get_queryset().exclude(completed=True).exclude(started=False)




    def get_completed(self) -> QuerySet:
        return super().get_queryset().filter(completed=True)




    def get_unclaimed(self) -> QuerySet:
        return super().get_queryset().exclude(started=True).exclude(completed=True)




    def get_successful(self) -> QuerySet:
        return super().get_queryset().filter(was_successful=True).filter(completed=T






class Ticket(models.Model):
    """
    The base class for tickets in the database. This is where the meat of the
    application purpose lies, as this table will hold all fields associated
    with a specific ticket. These fields are the professor, the section, the
    semester, the issue, the student who submitted the ticket, the tutor who
    primarily helped with the ticket, the tutor(s) who assisted the primary
    tutor, the name of the student, whether it was a successful ticket,
    the time the ticket was created, the date the ticket was created, the
    time the ticket was claimed (different than created), the time the
    ticket was closed, additional tutor comments, and if the ticket was
    reopened after it was closed.
    """

    NEW = "NEW"
    OPENED = "OPENED"
    CLOSED = "CLOSED"
```

```
    STATUS_CHOICES = [
        (NEW, "New"),
        (OPENED, "Opened"),
        (CLOSED, "Closed"),
    ]

    professor = models.ForeignKey("api.Professor", on_delete=models.PROTECT)
    # section = models.ForeignKey("api.Section", null=True, on_delete=models.PROTECT
    course = models.ForeignKey("api.Course", null=True, on_delete=models.PROTECT)
    issue = models.ForeignKey("api.Issues", null=True, on_delete=models.PROTECT)
    student = models.ForeignKey(
        "api.User",
        null=True,
        on_delete=models.PROTECT,
        related_name="student_ticket",
        blank=True,
    )
    tutor = models.ForeignKey(
        "api.User",
        null=True,
        on_delete=models.PROTECT,
        related_name="tutor_ticket",
        blank=True,
    )
    name = models.CharField(blank=False, max_length=50)
    title = models.CharField(blank=False, max_length=25)
    description = models.TextField(max_length=1024)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES, default=NEW)
    created_at = models.DateTimeField(null=False, blank=False, auto_now_add=True)
    opened_at = models.DateTimeField(null=True, blank=True)
    updated_at = models.DateTimeField(null=False, blank=False, auto_now_add=True)
    closed_at = models.DateTimeField(null=True, blank=True)
    was_successful = models.BooleanField(default=False)
    was_reopened = models.BooleanField(default=False)

    generic = models.Manager()
    ticket = TicketManager()

    def __str__(self) -> str:
        return self.name + " - " + str(self.course)
```