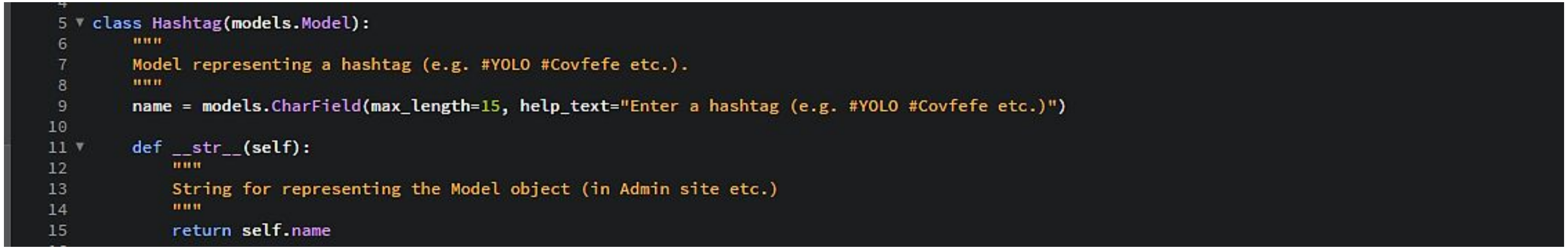
Data Models Diagram

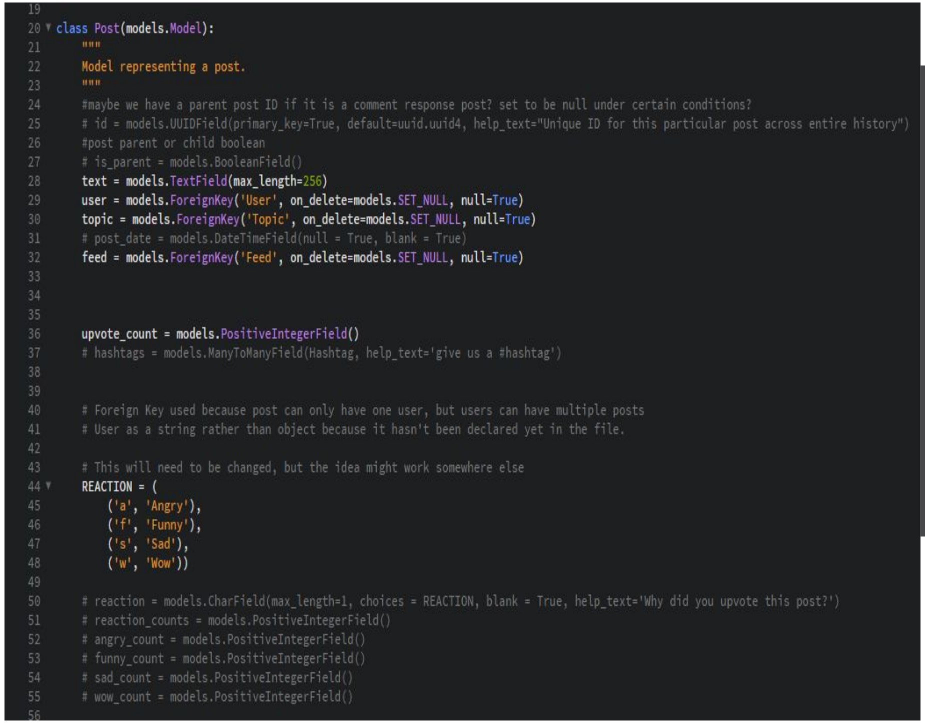
For “Venti”

Hashtag Model

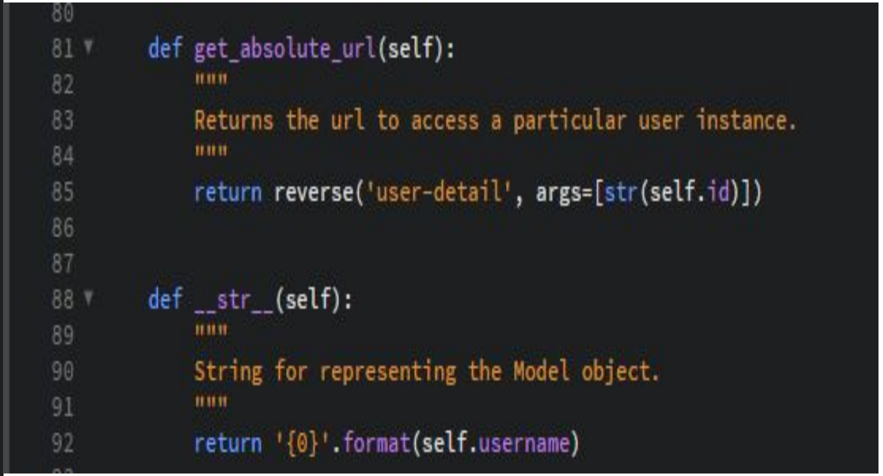


```
class Hashtag(models.Model):
    """
    Model representing a hashtag (e.g. #YOLO #Covfefe etc.).
    """
    name = models.CharField(max_length=15, help_text="Enter a hashtag (e.g. #YOLO #Covfefe etc.)")

    def __str__(self):
        """
        String for representing the Model object (in Admin site etc.)
        """
        return self.name
```

Post Model

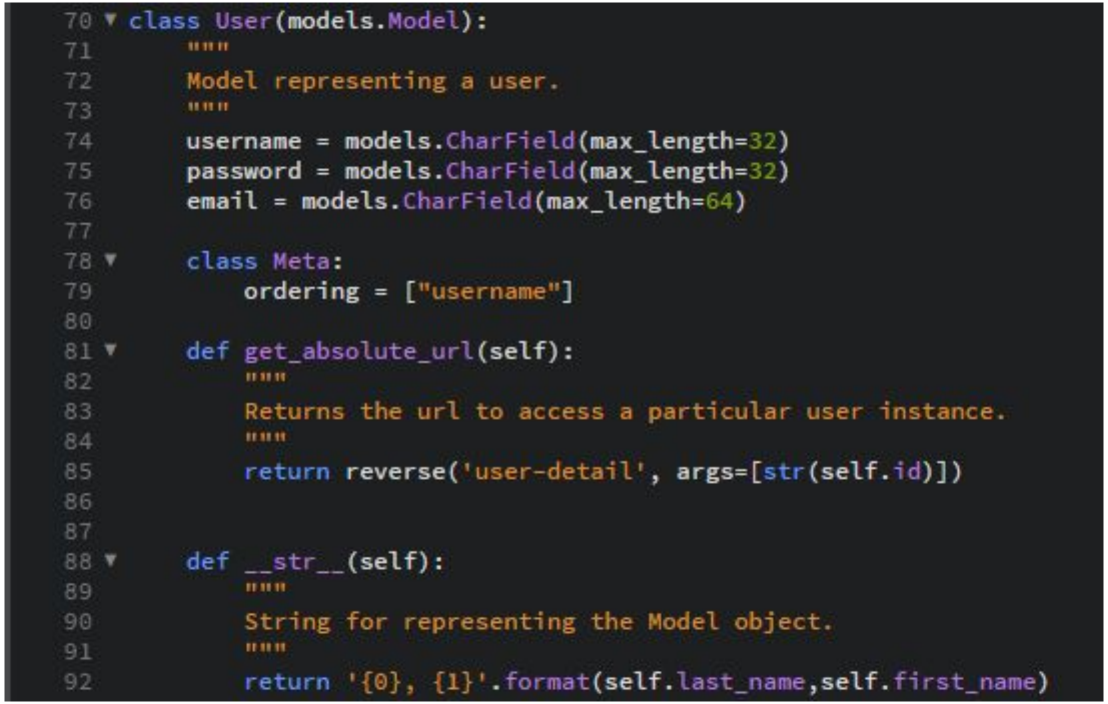


```python

class Post(models.Model):
    """
    Model representing a post.
    """
    #maybe we have a parent post ID if it is a comment response post? set to be null under certain conditions?
    # id = models.UUIDField(primary_key=True, default=uuid.uuid4, help_text="Unique ID for this particular post across entire history")
    #post parent or child boolean
    # is_parent = models.BooleanField()
    text = models.TextField(max_length=256)
    user = models.ForeignKey('User', on_delete=models.SET_NULL, null=True)
    topic = models.ForeignKey('Topic', on_delete=models.SET_NULL, null=True)
    # post_date = models.DateTimeField(null = True, blank = True)
    feed = models.ForeignKey('Feed', on_delete=models.SET_NULL, null=True)



    upvote_count = models.PositiveIntegerField()
    # hashtags = models.ManyToManyField(Hashtag, help_text='give us a #hashtag')


    # Foreign Key used because post can only have one user, but users can have multiple posts
    # User as a string rather than object because it hasn't been declared yet in the file.

    # This will need to be changed, but the idea might work somewhere else
    REACTION = (
        ('a', 'Angry'),
        ('f', 'Funny'),
        ('s', 'Sad'),
        ('w', 'Wow'))

    # reaction = models.CharField(max_length=1, choices = REACTION, blank = True, help_text='Why did you upvote this post?')
    # reaction_counts = models.PositiveIntegerField()
    # angry_count = models.PositiveIntegerField()
    # funny_count = models.PositiveIntegerField()
    # sad_count = models.PositiveIntegerField()
    # wow_count = models.PositiveIntegerField()

```



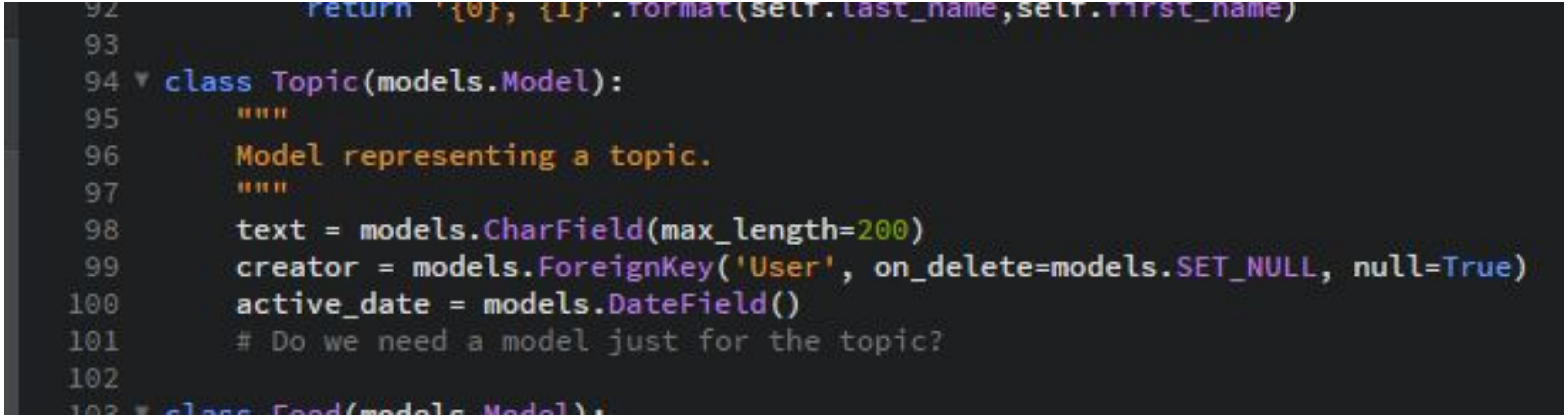
```python

    def get_absolute_url(self):
        """
        Returns the url to access a particular user instance.
        """
        return reverse('user-detail', args=[str(self.id)])


    def __str__(self):
        """
        String for representing the Model object.
        """
        return '{0}'.format(self.username)
```

User Model



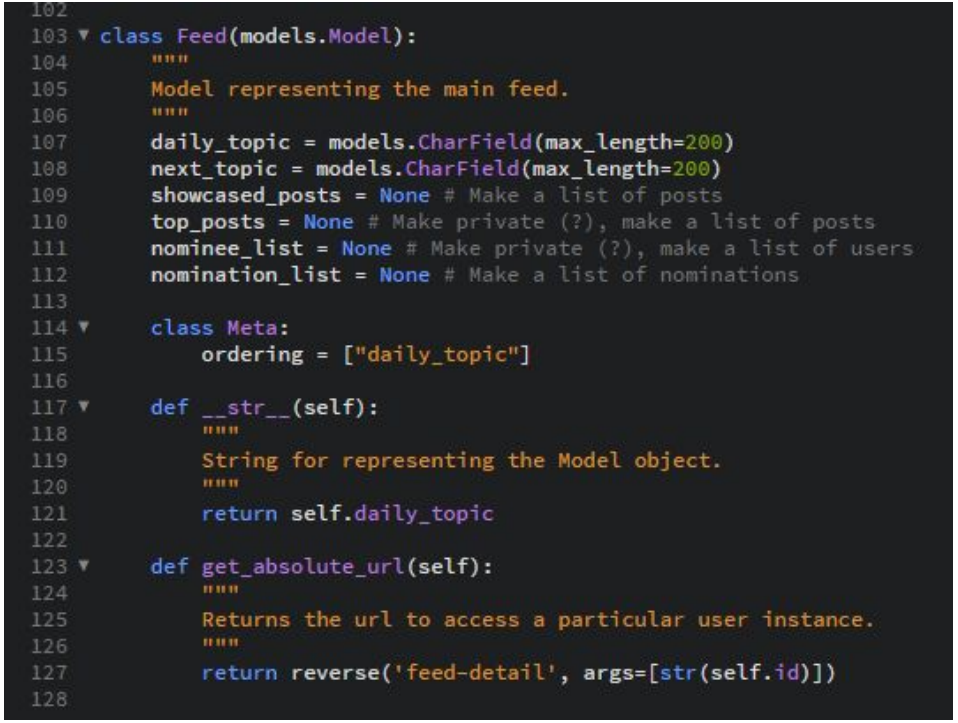
```python
class User(models.Model):
    """
    Model representing a user.
    """
    username = models.CharField(max_length=32)
    password = models.CharField(max_length=32)
    email = models.CharField(max_length=64)

    class Meta:
        ordering = ["username"]

    def get_absolute_url(self):
        """
        Returns the url to access a particular user instance.
        """
        return reverse('user-detail', args=[str(self.id)])


    def __str__(self):
        """
        String for representing the Model object.
        """
        return '{0}, {1}'.format(self.last_name,self.first_name)
```

Topic Model



```python
        return '{0}, {1}'.format(self.last_name,self.first_name)

class Topic(models.Model):
    """
    Model representing a topic.
    """
    text = models.CharField(max_length=200)
    creator = models.ForeignKey('User', on_delete=models.SET_NULL, null=True)
    active_date = models.DateField()
    # Do we need a model just for the topic?

class Food(models.Model):
```

Feed Model



```
class Feed(models.Model):
    """
    Model representing the main feed.
    """
    daily_topic = models.CharField(max_length=200)
    next_topic = models.CharField(max_length=200)
    showcased_posts = None # Make a list of posts
    top_posts = None # Make private (?), make a list of posts
    nominee_list = None # Make private (?), make a list of users
    nomination_list = None # Make a list of nominations

    class Meta:
        ordering = ["daily_topic"]

    def __str__(self):
        """
        String for representing the Model object.
        """
        return self.daily_topic

    def get_absolute_url(self):
        """
        Returns the url to access a particular user instance.
        """
        return reverse('feed-detail', args=[str(self.id)])

```