```
from django.db import models
from django.contrib.auth.models import User
from django.db.models import Q


class Repository(models.Model):
        creator = models.ForeignKey(ProfileUser)
        name = models.CharField(max_length=200)
        contributors = ManyToManyField(ProfileUser, related_name="contributed", blank=True)
#sorted by number of commits descending order
        language = ManyToManyField(Language, related_name="used_in", blank=True)
        #number next to each lang is number of bytes of code in that language
        stack = ManyToManyField(Stack, related_name="used_in", blank=True)
        tags = models.ManytoManyField(Tag, related_name=”tagged_as”, blank=True)
        #commitActivityData = array of array of ints, may want to save this
        date_created = models.DateTimeField(auto_now_add=False)
        star_count = models.IntegerField()
        fork_count = models.IntegerField()
        kb_size = models.IntegerField()

class Comment(models.Model):
        profile_user = models.ForeignKey(ProfileUser)
        body = models.CharField(max_length=400) #text of comment
        repository = models.ForeignKey(Repository)
        date_time = models.DateTimeField(auto_now_add=True)
        path = models.CharField(max_length=400) #relative path file comment on
        likes = models.IntegerField()
        post = models.ForeignKey(Post)

class ProfileUser(models.Model):
        user = models.OneToOneField(User)
        following = models.ManyToManyField("self", related_name="followers", blank=True)
        #can list followers of user and list users followed by another user
        rating = models.ManyToManyField(Rating, related_name="rated", blank=True)
        email = models.CharField(max_length=400)
        website = models.CharField(max_length=400, blank=True)
        company = models.CharField(max_length=400, blank=True)
        bio = models.CharField(max_length=400, blank=True)

class Rating(models.Model):
        language = models.ForeignKey(Language)
        proficiency = models.IntegerField()    # how well they think they know the language
        credibility = models.IntegerField() # how well we think they know the language
```

```
class Language(models.Model):
        name = models.CharField(max_length=20)
        extensions = models.CharField()

class Stack(models.Model):
        name = models.CharField(max_length=40)

class Difficulty(models.Model):
        rating = models.IntegerField()
        repository = models.ForeignKey(Repository)
        profile_user = models.ForeignKey(ProfileUser)
        date_time = models.DateTimeField(auto_now_add=True)

class Tag(models.Model):
        message = models.CharField(max_length=40)
        profile_user = models.ForeignKey(ProfileUser)
        repository = models.ForeignKey(Repository)
        date_time = models.DateTimeField(auto_now_add=True)
        endorsements = models.ManyToManyField(ProfileUser)
        #tagger field with hash info about tagger - name, email, date

class Watch(models.Model):
        #can list watchers and list repositories being watched
        profile_user = models.ForeignKey(ProfileUser)
        repository = models.ForeignKey(Repository)
        date_time = models.DateTimeField(auto_now_add=True)

class Star(models.Model):
        #can list stargazers and list repositories being starred
        profile_user = models.ForeignKey(ProfileUser)
        repository = models.ForeignKey(Repository)
        date_time = models.DateTimeField(auto_now_add=True)

class Post(models.Model):
        profile_user = models.ForeignKey(ProfileUser)
        rating = models.ManyToManyField(Rating, related_name="rated", blank=True)
        language = ManyToManyField(Language, related_name="used_in", blank=True)
        tags = models.ManytoManyField(Tag, related_name=”tagged_as”, blank=True)
```