README

Adam Cole
04/26/2019

TeleSign Coding Challenge

Models.py

- I created four classes in order to implement the Menu Inventory

class MenuItem(models.Model):
- superclass of Drink and Snack
- made a superclass so Ingredient could be foreign key related to both Snacks and Drinks independently
- string variable to hold if snack or drink

class Drink(MenuItem):
- subclass of MenuItem
- string variable to hold snack name

class Snack(models.Model):
- subclass of MenuItem
- ManyToMany relation to Drink to implement recommended snack and drink pairs
- string variable to hold drink name

class Ingredient(models.Model):
- Foreign Key relation to MenuItem (so both Drink and Snack)
- string variable to hold ingredient name

My Submission

- Unable to Develop the backend of the add/edit/delete options of the menu interface before the deadline.
- I failed to store the input from the Django form into variables in python code
- Selecting the menu items and displaying properties of the menu item are done.
- To add Snacks or Drinks, must be done in the $ python3 manage.py shell

(example)

```
>>> from menu.models import Drink, Snack, Ingredient
```

(creating Drinks)

```
>>> d1 = Drink(drink_name="espresso", item="drink")
>>> d2 = Drink(drink_name="espresso", item="drink")
>>> d1.save()
>>> d2.save()
```

(Adding ingredients //foreign key)

```
>>> d1.ingredient_set.create(ingred_name="espresso beans")
```

```
>>> d1.ingredient_set.create(ingred_name="milk")
>>> d1.ingredient_set.create(ingred_name="foam")
>>> d2.ingredient_set.create(ingred_name="espresso beans")
>>> d2.ingredient_set.create(ingred_name="milk")
>>> d2.ingredient_set.create(ingred_name="hot water")
```

(creating Snacks)

```
>>> s = Snack(snack_name="croissant", item="snack")
>>> s.save()
>>> s.ingredient_set.create(ingred_name="flour")
>>> s.ingredient_set.create(ingred_name="butter")
>>> s.ingredient_set.create(ingred_name="milk")
>>> s.ingredient_set.create(ingred_name="yeast")
```

(adding Snack and Drink recommended pair //many to many key)

```
>>> d1.snack_set.add(s)
>>> d2.snack_set.add(s)
```

(printing relations)

```
>>> d1.snack_set.all()
>>> d2.snack_set.all()
>>> s.reccomend.all()
```

What I would do differently the second time around:

- *Use generic/template views instead of definitions for all my methods*
- *switch the meaning of MenuItem's string variable and the subclass's string variable to have MenuItem hold the name of the item, and have the two subclasses hold whether snack or drink*
- *Figure out how to utilize GET and POST to store data from Django forms into my python code*
- *Create unit tests to run as I go, not wait until the end of development*

Notes about Development:

- *Used Django 2.2 instead of 2.0 since it was no longer supported*
- *Used MySQL for backend development*