COMP 8347: Internet Applications and Distributed Systems

FALL 2020 LAB #9

sPART 1: Work with ModelForms

1. Create form class. In the file *myapp/forms.py* and add the following lines:

```
from django import forms    #This line may already exist in your file
from myapp.models import Order
```

a. Create a new form class: **`class OrderForm(forms.ModelForm):`**

b. *OrderForm* should be a form based on the **Order** model. The form fields should include: *courses*, *student*, and *order_status*

c. Add the following lines in your *forms.py* file to create the new class

```
    class OrderForm(forms.ModelForm):
        class Meta:
            model = Order
            fields = ['courses', 'student', 'order_status']
            widgets = {'courses': forms.CheckboxSelectMultiple(),
    'order_type':forms.RadioSelect}
            labels = {'student': u'Student Name', }
```

2. Create *place_order* view. This view will display a form that allows a user to place a new *order* for course(s).

a. Edit your *views.py* file by adding the following function:

```
def place_order(request):
    if request.method == 'POST':
        form = OrderForm(request.POST)
        if form.is_valid():
            courses = form.cleaned_data['courses']
            order = form.save(commit=False)
            student = order.student
            status = order.order_status
            order.save()
            if status == 1:
                for c in order.courses.all():
                    student.registered_courses.add(c)
            return render(request, 'myapp/order_response.html', {'courses':
courses, 'order':order})
        else:
            return render(request, 'myapp/place_order.html', {'form':form})

    else:
        form = OrderForm()
        return render(request, 'myapp/place_order.html', {'form':form})
```

b. Create the template *place_order.html* in *myapp/templates/myapp* dir. This template should render the form you created in *place_order* view.

c. Also, create the template *order_response.html* to display the title(s) of the *course*(s) in the *order* that was entered.

d. Update *myapp/urls.py:* Add the necessary *path* to *myapp/urls.py* file so this *place_order* view function is executed if the user accesses the url *myapp/place_order* under your website

e. Update *index.html* to add a link to *place_order* page using **url namespacing** tag.

3. Create and check new order:
a. Create a new *order* and add it to the database, using the above form, view function and templates.
b. Check the new order in the admin interface. Are the selected courses correctly associated with the order?
c. If not, modify the above view function so that the selected courses associated with the order will be correctly saved in the database.

PART 2: Create a new ModelForm

1. Update **Course** model.

a. Edit *models.py* to add a new *PositiveIntegerField* called *num_reviews* to the **Course** model. This field indicates how many reviews have been submitted for this *course*. The default value for the field is **0**.

b. Update database. (Use makemigrations, sqlmigrate, migrate)

2. Create a new model.

a. Edit *models.py* to add the **Review** model as follows.

```
class Review(models.Model):
    reviewer = models.EmailField()
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    rating = models.PositiveIntegerField()
    comments = models.TextField()
    date = models.DateField(default=timezone.now)
```

b. Add a suitable **__str__** method for this model. Make **comments** optional.
c. Update database. (Use makemigrations, sqlmigrate, migrate)

3. Create new forms. In your file *myapp/forms.py* import any necessary models then create a new class as follows:

a. Create a new form class: **`class ReviewForm(forms.ModelForm):`**

b. `ReviewForm` should be a form based on the **`Review`** model. The form fields should include: the fields *reviewer*, *book*, *rating*, and *comments*

c. Set the widget for *book* field to **RadioSelect**.

d. Set the label for the *reviewer* field to '**Please enter a valid email**' and the label for the *rating* field to '**Rating: An integer between 1 (worst) and 5 (best)**'

4. Create *review* view function. This function will provide a form for the user to review a selected book.

a. When the user goes to url `myapp/review` the function *review(request)* should display an empty form (i.e. ReviewForm) for the user.

If a *valid* form is submitted:

i) Check if the *rating* is between 1 and 5. If so, a

a new **Review** object is created based on the information submitted and stored in the db

the num_reviews field of the specified book is incremented by 1 and the updated value is saved in the db.

the user is redirected to the main (*index.html*) page.

ii) if the *rating* is not between 1 and 5:

Redisplay the form with the message: 'You must enter a rating between 1 and 5!'

If the submitted form is not valid, appropriate error messages are displayed to the user.

b. Create the template *review.html* in *myapp/templates/myapp* dir to display the ReviewForm.
c. Update *myapp/urls.py:* Add the necessary *path* to *myapp/urls.py* file so this *review* view function is executed if the user accesses the url *myapp/review* under your website