# Database Heaven

**Kalu Kalu**

Database Heaven 1
Test Description 3
Client Specifications 3
Review Will Be Almost Exactly The Same as The Test 3
Rubric: 3
Points Deduction 4
Kyrie's Blog 4
Creating Our models.py 5
We Must makemigrations and migrate 6
Editing Our views.py 7
For A POST Request 7
For A Get Request 8
Template For-Loops In home_page.html 8
Runserver And Checkout Website 10
Setting Up Admin.py 10

# Test Description

The home tests the student's ability to make a web-app, and share it open-sourced on Github.

For the test exam the students will be asked to launch their terminal, activate their virtual environment and accomplish the following tasks:

1) Use Django to create a simple proof of concept web-app for our client, a barbershop store, based on the client specifications
2) Upload the entire Django project onto Github

## Client Specifications

1. Title says "Amazing Amy's Barbershop"
2. Webpage header should have a large heading 1 with the retail store name, Amazing Amy's Barbershop
3. Below the barbershop name should be the company tagline, "A Fresh Cut You And Your Family!" in smaller font. (you can use your own tagline but it must be catchy!)
4. There should be pictures of at least 3 barber as well as their name of the item.
5. The website should have a working form that allows users to submit the name of the barber they want, the own customer's name, as well as their appointment time.
6. After a user submits their form they should see the information they submitted printed on the webpage below the order form.
7. **Mandatory:** Save all user inputs in a database in a table named appointment
8. **Mandatory**: Create a user admin where we can see data in our database.

## Review Will Be Almost Exactly The Same as The Test

This review will be almost exactly the same as what the test exam will ask you. The only difference will be the name of the app, the name of the project, and some text. Instead of making a convenient store website for "Kool Karen Retail Store", where users can post what item they want and their address, you might instead be asked to make an online ice cream store website for "Evelyn Eats Ice Cream Shop" where users post which ice cream they want and what their address it.

Other than that it will be exactly the same.

# Rubric:

The students must take a screenshot of their Terminal and save it in their ~/Documents/your_name folder. Immediately after the exam I will look at the pictures and grade their exams.

The exam will be graded based on the following rubric.

1) Django Project (70)
    1) Have a working website which can be visited using the 127.0.0.0:8000 web address. (10 points)
    2) Include the proper webpage (specification 2) (10 points)

3) Create your database table named Appointment (specification 4) (10 points)
4) Include a working form in home_page (specification 5) (10 points)
5) Everybody's previous appointments is listed at the very bottom (10 points)
6) When user submits they can see their output as well as everybody else previous appointments (specification 6) (10 points)
7) Create Your admin.py (10 points)

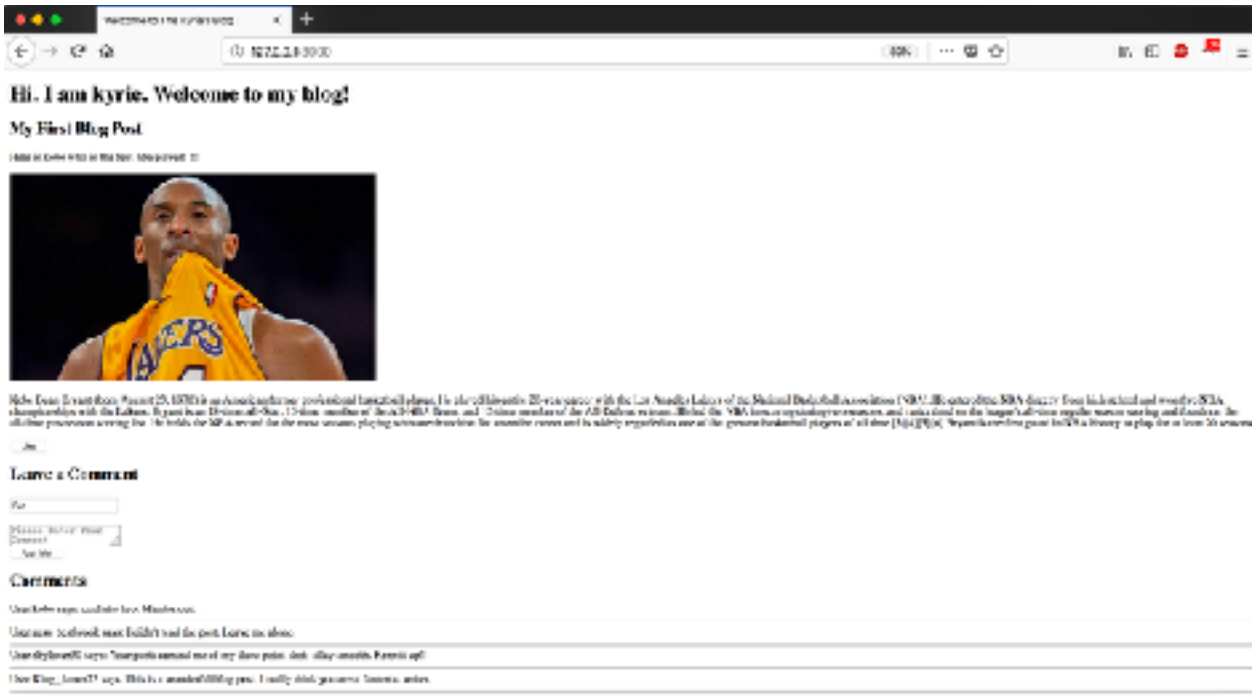
2) Upload to Github (30)
   1) Create a local repository
   2) Create a GitHub repository
   3) Save all changes to local repository and push to remote repository.

## Points Deduction

I will deduct 5 points off your score for any time you need my help to progress or if you speak to another classmate (which is not allowed).

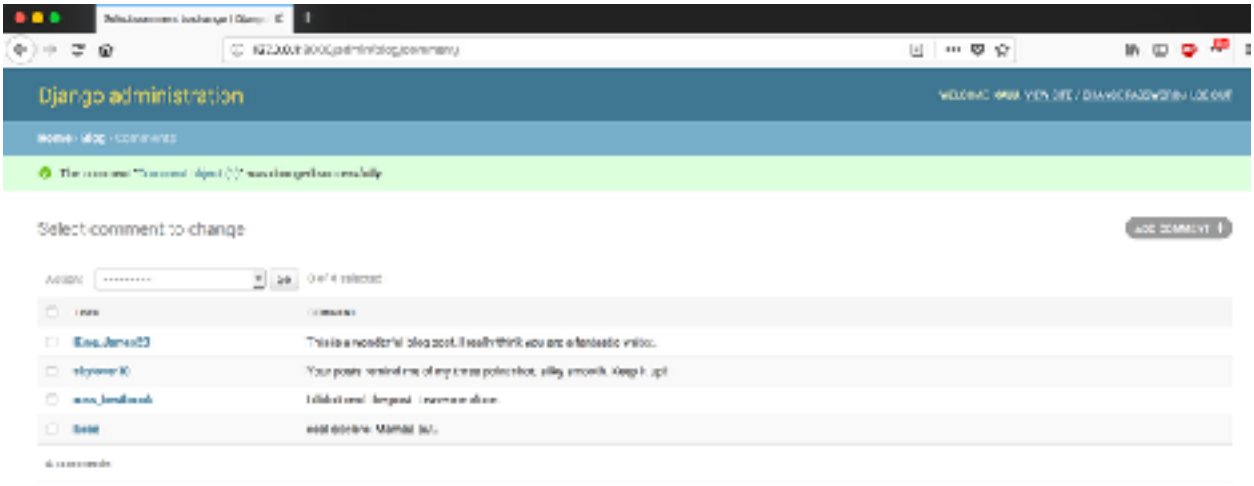
# Kyrie's Blog

Our good friend Kyrie has been making some progress on his blog. Why don't we stop by and see how he's doing.




Whew! That's a nice looking website! Notice that comments left by old users are now being stored and read from the database. Wonderful!

But that's not all Kyrie has done. He has even gone so far as to hook up an admin interface to his database. Let's take a look!




We can access this admin interface to our database through the http://127.0.0.1:8000/admin url. From here we can directly click on any app table and CRUD ( create, read, update, and delete) any row of data saved in the table.

Tough work Kyrie! Now let's learn how he did all of this.

## Creating Our models.py

Let's create our models.py in case you haven't already.  Here is Kyrie's models.py.

*models.py*

```
1. from django.db import models
2.
3. # Create your models here.
4. class Comment(models.Model):  1
5.     comment = models.CharField(max_length=180, null=True, blank=True)  2
6.     user = models.CharField(max_length=180, null=True, blank=True)  3
```

The important part is the definition of our Comment class. Django creates models by creating a class that inherits from models.Model. Therefore when create our Comments table, we simply have to define a class with models.Model as a class parameter like so, class

Comment(models.Model) 1 . Next we want to create the attributes for our model. Our attributes as similar to column headers. We decide to create a comment attribute 2 , and user attribute , 3 .

We can define attributes by calling one of several field methods on the models module. In both location 2 and 3 we called the models.CharField() method and passed the following keyword arguments, max_length, null, and blank, each with the values 180, True, and True. max_length=180 means that our comment or user value in our database table cannot be longer than 180 characters. null=True means that a database record can have no comment at all. And lastly, blank=True means that the value for that respective attribute can simply an empty string “”.

## We Must makemigrations and migrate

Anytime we make a change to models.py, we must makemigrations and migrate so that the changes are reflected on our database.

Open your terminal, activate your virtual environment, and cd into your project folder where manage.py is located.

You should type the following code.

```
(django_and_machine_learning_venv) KaludeMacBook-Pro:your_repo_url_address kalukalu$ python manage.py makemigrations
```

You should get the following output.

```
Migrations for 'blog':
  blog/migrations/0001_initial.py
    - Create model Comment
```

Next we can migrate.

```
(django_and_machine_learning_venv) KaludeMacBook-Pro:your_repo_url_address kalukalu$ python manage.py migrate
```

And again you should get the following output:

```
  Applying blog.0001_initial... OK
```

Congrats! We have created our database table, Comment.

Let’s now move on to our views.py. In our views, we want to be able to save our comments to our newly created database table. Remember, anytime we’re talking about working with

databases, we want to think CRUD, Create, Read, Update, or Delete. Next we want to create a database record in views.py.

## Editing Our views.py

Let's look at Kyrie's views.py.

*views.py*

```
1.  from django.shortcuts import render
2.  from . models import Comment
3.
4.
5.  def home_page(request):
6.      if request.method == 'POST':
7.          our_comment = request.POST.get('comment')
8.          our_user = request.POST.get("username")
9.          comment_obj = Comment.objects.create( 1
10.             comment=our_comment, 2
11.             user=our_user 3
12.         )
13.         all_comments = Comment.objects.all() 4
14.         context = {'comments': all_comments} 5
15.
16.         return render(request, 'blog/home_page.html', context) 6
17.
18.     if request.method ==  "GET": 7
19.         all_comments = Comment.objects.all() 8
20.         context = {'comments': all_comments}  9
21.         return render(request, 'blog/home_page.html', context) 10
```

---

## For A POST Request

In position 1 we import our Comment model class from models.py. In line number 9 is the first part we where change our views. In the past we got our comment and username from the request.POST.get("comment") and request.POST.get("username") and saved it the variable names our_comment and our_user. We do the same thing as before, but now we do not send those variables back to our template using the render function. Instead we use the comment and user variables to create a row/record in our Comment database table.

We create a record by calling Comment.objects.create() and passing the create function keyword arguments we created as attributes our models.py Comment class. Because our Comment class has comment and user as attributes, we need to pass create function a comment=our_comment and 2 and user=our_user 3, argument-value combination. There

we call  Comment.objects.create(comment=some_value, user=some_user) and save the result of calling that function to the variable name “comment_obj” 1 .

Now that we have (**C**.R.U.D) **CREATED** our record, we want to **READ** (C.**R**.U.D) every row in our database table. That way we have ALL comments every created. We do that at location 4 using the code Comment.objects.all(), which grabs us a list like object containing all our comment records. Then we save it in the all_comments variables.

Lastly, we want to create a context dictionary which we will pass to our render function 5 . The context dictionary is data that we want to have available to our template. We want to be able to print all saved comments in our template, so we must pass our all_comments list to our context dictionary and pass that to our render function as the third argument 6 . Now inside of our view we will have access to our comments list using the keyword “comments”. We simply need to use double curly bracket notation to use it.

---

## For A Get Request

Next we set up an if condition block that will run when our user makes a GET request 7 . Just like we did in our POST request if statement block, we merely want to save all our comment records from our Comment table into a variable called all_comments 8 . Then we create a context dictionary 9 and return that context dictonary to our template as the third argument of our render function 10 .

Next, we will set up our home_page.html to display all the comments passed to it through the context dictionary.

# Template For-Loops In home_page.html

Next, we need to create a for-loop, a way to loop through all the elements in a list. Remember that we have this list like object, “comments”, that contains a collection of every record in our database. We can display all comments by using Django template syntax for for-loops. The syntax goes like this {% for variable_name in list_object %} variable_name.attribute {% endfor %}. Here instead of list_object we will use our “comments” object that was passed to our template in a context dictionary from our view render functions. And we will choose a variable name of our own creation that we will use to later reference each comment in our comments list. We can choose the name comment, because that what it is, a comment. And lastly, our attributes we want to access (aka print out) are comment and user, so we would need to do comment.user and comment.comment to get access to each comment records user and comment attributes’ values. Therefore we may want to do something like {% for comment in comments %} comment.user - comment.comment {% endfor %}

Let’s see the code below.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.  <title>Welcome to The Kyrie's Blog</title>
5. </head>
```

```
<body>
    <!-- Site Header -->
    <h1>Hi. I am kyrie. Welcome to my blog!</h1>

    <!-- Blog Post -->
    <h2>My First Blog Post</h2>
    <p>Here is kobe who is the best nba player!!!!</p>
    <p><img src="https://ss0.bdstatic.com/70cFuHSh_Q1YnxGkpoWK1HF6hhy/it/u=1605251468,2192982596&fm=27&gp=0.jpg" </p>
    <p>Kobe Bean Bryant (born August 23, 1978) is an American former professional basketball player. He played his entire 20-year career with the Los Angeles Lakers of the National Basketball Association (NBA). He entered the NBA directly from high school and won five NBA championships with the Lakers. Bryant is an 18-time All-Star, 15-time member of the All-NBA Team, and 12-time member of the All-Defensive team. He led the NBA in scoring during two seasons, and ranks third on the league's all-time regular season scoring and fourth on the all-time postseason scoring list. He holds the NBA record for the most seasons playing with one franchise for an entire career and is widely regarded as one of the greatest basketball players of all time.[3][4][5][6] Bryant is the first guard in NBA history to play for at least 20 seasons</p>

    <!-- Social  -->
    <button>Like</button>

    <!-- Leave A Comment -->
    <h2>Leave a Comment</h2>
    <form action="{% url 'blog:home_page' %}" method="POST">
        {% csrf_token %}
        <input type="text" placeholder="Please Enter Your username" name="username"><br><br>
        <textarea name="comment" placeholder="Please Enter Your Comment"></textarea><br>
        <input type="submit" value="Post Me!">
    </form>

    <!-- Display Previous Comments -->
    <h2>Comments</h2>
    {% for comment in comments %}
        User {{comment.user}} says: {{comment.comment}}<hr>
    {% endfor %}
</body>
</html>
```

## Runserver And Checkout Website

Now we can runserver and visit our localhost url address.

```
(django_and_machine_learning_venv) KaludeMacBook-Pro:your_repo_url_address kalukalu$ python manage.py runserver
```

This should give us the following output

```
Performing system checks...

System check identified no issues (0 silenced).
May 02, 2018 - 09:29:53
Django version 2.0.4, using settings 'my_site.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Now we simply need to go to our 127.0.0.1:8000/ address and see our wonderful website.

## Setting Up Admin.py

Now we want to set up admin.py. Open your blog/admin.py and type the following inside of it.

```
1. from django.contrib import admin
2. from .models import Comment  1
3.
4.
5. # Register your models here.
6. class CommentAdmin(admin.ModelAdmin):
7.     list_display = ['user',  'comment']  2
8.
9. admin.site.register(Comment, CommentAdmin)
```

For now use write this code. The important part is that we need to import our Comment model from models.py 1 and then we need to create a list titled “list_display” containing the names of our Comment model attributes as setup in models.py 2 .

After this we can now go to 127.0.0.1:8000/admin url in our browser and checkout our awesome new admin. But first, let’s create a superuser. In your Terminal first kill your server using Ctrl+C and then type the command python manage.py createsuperuser to create your

superuser. You will be asked to type a username and password. Choose a simple one you can remember.

```
Quit the server with CONTROL-C.
^C(django_and_machine_learning_venv) KaludeMacBook-
Pro:your_repo_url_address kalalu$ python manage.py createsuperuser
Username (leave blank to use 'kalukalu'): kyrie
Email address: kyrietheballa@qq.com
Password:
Password (again):
Superuser created successfully.
```

And now finally check out your admin,  at this url, 127.0.0.1:8000/admin

Log in with the username and password you just created.