**Objective 4 Report**

**Objective 4 - Users can share some form of multimedia content which is stored and hosted on your server**

* *Describe how you completed the objectives of the assignment, how your software is structured, and why you made the design decisions you made. A reader should have a good understanding of your software by reading this section without looking at a single line of your code*

**/forums/0001\_initial.py**

This file is related to Django’s engine called ORM Object Relational Mapper which translates python code into SQL code. Django’s ORM provides a way for the data in Django’s database to be accessed. All of the code here is created by Django. These are the “initial migrations” for our application that are following the order of how our model.py is structured.

Libraries and their links :

**Settings** -> We used AUTH\_USER\_MODEL

<https://docs.djangoproject.com/en/3.2/ref/settings/>

Models are used by Django to store an application’s data. This is used to return a string that will give the position of the User Model ( ‘auth.User’ which will then call get\_use\_model() to return the user model class that is specified. The user model contains fields that will be referencing the other models that are in our application. We are using this because it’s involved with migrations which is Django’s way of being able to make changes to our models such as adding fields, deleting models in our Django database.

**/Migrations** ->

<https://docs.djangoproject.com/en/3.2/topics/migrations/>

<https://github.com/django/django/blob/main/django/db/migrations/migration.py>

In our case, we are using migrations library and the function swappable.dependencies. We have to use this because Django creates a special key with the User Model and the AUTH\_USER\_MODEL and the relationship of each migration will be a reference to the settings in the AUTH\_USER\_MODEL and not the model directly. Django uses models to create tables, their fields and constraints within each model. Each model that we are creating with the function CreateModel function from the migrations library.

**CreateModel** ->

<https://docs.djangoproject.com/en/3.2/ref/migration-operations/>

Django uses models to create tables, their fields, and constraints associated with a model to store user data. Each model maps to a single database table. With this library, we are using the Create Model to create a table in the database to match with the fields given.

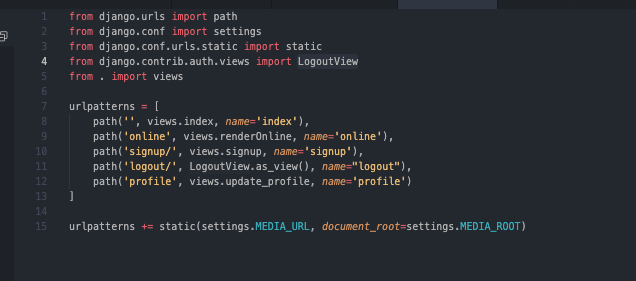
**Django/db/models/deletion (Used Set\_NULL)** Line 58 ->

<https://github.com/django/django/blob/main/django/db/models/deletion.py>

This function works by deleting the referenced object it corresponds to. So if it was on ‘updated\_by’ it would set the reference to NULL. It will replace the referenced object with “NULL” which is what SQL uses. Django will store the NULL in place of what was previously referenced there. This acts sort of like “deleting the data” in a more secure manner to allow for searches to be “User not found” instead of an error.

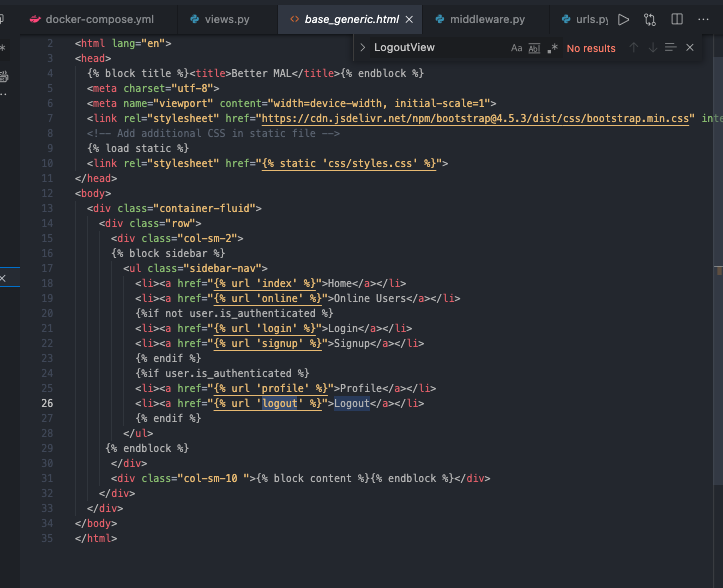
**/forum/urls.py**

*\* Path library is already talked about previously*



**What is happening here?**

Django utilizes the use of views which is a web request that generates some sort of web response. In our example, we are using the library path which is taking in a (route,view,kwargs,name). A route is the path that we want to have things rendered to. The view is a function that we have created in another document called view.py and the name is a custom name for a url pattern. A url pattern name is used so that our html will render the correct url in our templates.

****

**Libraries Used :**

**django/contrib/auth/views/logoutView :**

[**https://github.com/django/django/blob/7582d913e7db7f32e4cdcfafc177aa77cbbf4332/django/contrib/auth/views.py#L123**](https://github.com/django/django/blob/7582d913e7db7f32e4cdcfafc177aa77cbbf4332/django/contrib/auth/views.py#L123) Line : 111

* This view is from the django library which will render a logged\_out html page for the user when they click log out. It uses a variable called REDIRECT\_FIELD\_NAME which is bound in django.contrib.auth which binds REDIRECT\_FIELD\_NAME to ‘next’. ‘Next’ is used to specify the URL to be redirected to after the page. Since for logout it is NONE, then it will redirect to the logout.html we have in the template

**django/conf/urls/static.py :**

[**https://github.com/django/django/blob/7582d913e7db7f32e4cdcfafc177aa77cbbf4332/django/conf/urls/static.py**](https://github.com/django/django/blob/7582d913e7db7f32e4cdcfafc177aa77cbbf4332/django/conf/urls/static.py)Line: 10

* This returns a URL pattern for serving files in debug mode. We didn’t really use this because it’s associated for images during deployment aka for the admin
* This static library is used for when we are serving static files in development, because when we use the settings library and append MEDIA\_URL, we want to save the files for deployment.

**django/conf/settings.py /** *Not a library but is involved with settings.py*

* We are using these libraries for MEDIA\_ROOT and MEDIA\_URL
* MEDIA\_ROOT is also for stored files. MEDIA\_ROOT is a filesystem path to the directory that will hold all user-uploaded files. Kind of like the master branch while media\_url is just a branch.
* MEDIA\_URL is a url under MEDIA\_ROOT that handles stored files. These files are configured during deployment.

**/forum/apps.py**

**Github Library :**

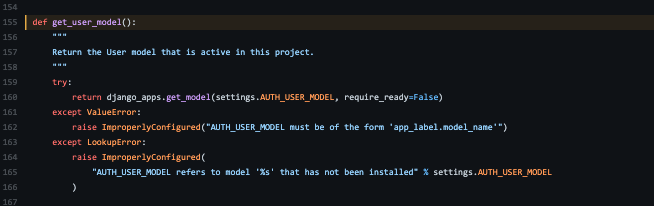
[**https://github.com/django/django/blob/main/django/contrib/auth/apps.py**](https://github.com/django/django/blob/main/django/contrib/auth/apps.py)

**More Documentation :** [**https://docs.djangoproject.com/en/3.2/ref/applications/**](https://docs.djangoproject.com/en/3.2/ref/applications/)

The purpose of this file is to help developers include any configuration needed when creating a Django project. This way, when our application loads, AppConfig will specify our app to be “forum” so that when our INSTALLED\_APPS are called, we can make use of the name “forumConfig” without having to change the INSTALLED\_APP settings manually. This is used to promote easier configuration by changing the class in the INSTALLED\_APPS settings.

**/forums/forms.py**

This part of our project deals with the database and how Django utilizes forms as a container for the user to send data to our application. Django provides a helper class that allows us to create a Form class from an already finished Django mode. Fields are attributes that are associated with each model. If you look at our code, we have a UserForm implemented from the Django library .forms. We are creating these forms based on the models that we have created ourselves in models.py

**Library Used: **

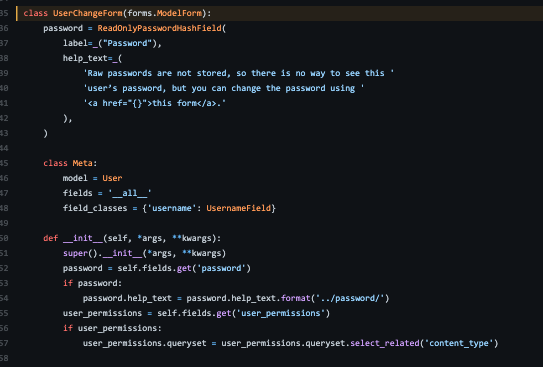
**Django/contrib/auth/get\_user\_model :**

[**https://github.com/django/django/blob/c240ceea7d88c6a8058dcacb37356c93e0a3618f/django/contrib/auth/\_\_init\_\_.py#L155**](https://github.com/django/django/blob/c240ceea7d88c6a8058dcacb37356c93e0a3618f/django/contrib/auth/__init__.py#L155)

This library is used because we need to do an authentication process before we can retrieve the User’s profile. We will be using the classes that we make in this file to be used in views.py later on. We are using the get\_user\_model to reference the current active model so that we can use it later for views.py. The UserForm and ProfileForm are used to be able to change a user’s information and their permissions.

**Django/contrib/auth/forms/UserChangeForm**

[**https://github.com/django/django/blob/c240ceea7d88c6a8058dcacb37356c93e0a3618f/django/contrib/auth/forms.py#L135**](https://github.com/django/django/blob/c240ceea7d88c6a8058dcacb37356c93e0a3618f/django/contrib/auth/forms.py#L135)

****

UserChangeFrom is a class that enables the admin to change a user’s information and their permissions. We are going to be using this to make our own custom forms from the models that we have created in models.py.

**Django/forms/ModelForm**

This library is used because we are going to be creating the form in our forms file, but in our models, we’re going to label what would be a “User” and what would be a “Userprofile”.

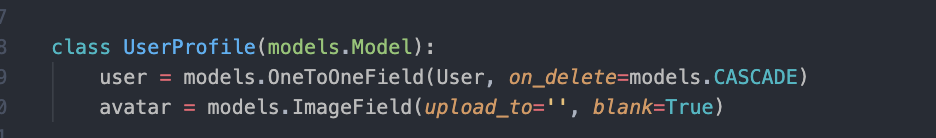
**/forums/models.py**

Django uses models to maintain data stored in its database that is received from the user to our application. These models are defined in our models.py and they use the django.db models library to create subclasses (classes derived from a parent class with the same functionalities) so that we can customize our own models based on what we want them to contain.

**Django/db/models**

**Github Documentation:** [**https://github.com/django/django/blob/c240ceea7d88c6a8058dcacb37356c93e0a3618f/django/db/models/fields/related.py#L1027**](https://github.com/django/django/blob/c240ceea7d88c6a8058dcacb37356c93e0a3618f/django/db/models/fields/related.py#L1027)Line 1027

If you look at our code : Here we are creating a model defined as “UserProfile” and which will contain 2 fields which are user and avatar which will be an image. For avatar, the blank set to true allows for the field to be allowed to be blank. By default, django would set that blank to false and the field will be required instead of optional. Cascade is used to tell Django to delete all related records when a record is deleted - an example of this would be if we were to delete an album, then then all the songs in the album would be deleted as well. In this case if the user is deleted then we would delete the record. This is different from NULL where Django will store empty values instead of deleting.

****

In our code, we are also calling the function OneToOneField which is a field used when one record of a model is related to another model to allow for that field to be a “key” to the object. We can use UserProfile as an example. Just like how a car has a one-to-one relationship with it’s license plate, our relationship with a model called UserProfile has a one-to-on User so each UserProfile is associated with a specific User.

**django/db/models/signals/post\_save**

Github documentation : <https://github.com/django/django/blob/c240ceea7d88c6a8058dcacb37356c93e0a3618f/django/db/models/signals.py>

Django uses signals to allow the application to be notified when certain events occur. In order for signals to work, there has to be 2 elements in the parameters. There has to be a sender who is responsible for initiating the signal and there has to be a receiver who will be the one that receives the signal and does something. In our case, we are going to be putting the function create\_profile as the receiver that will receive the signal and we are going to be putting the sender as the User and create a new instance of a UserProfile. So each time the User initializes the save method then create\_profile will be executed. Post\_save is the signal and it is going to be calling ModelSignal and use the connect function to do the connection between the senders and receivers through instances of Signal. Create\_profile is the receiver function, post\_save is the signal and we will be creating instances of connections.

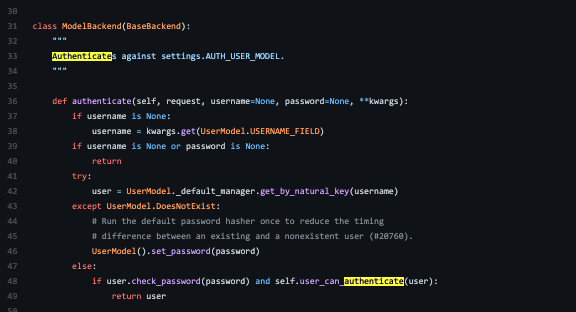
**/forums/views.py**

Views are python functions that receive a HttpRequest object and return an HttpResponse object. Receive a request as a parameter and a response is sent as a result. In this file we are showing how many users are currently online, we are also allowing for updates in our profile, the ability to sign up, and log out

**django/contrib/messages ->** did not use

**django/contrib/auth/authenticate**

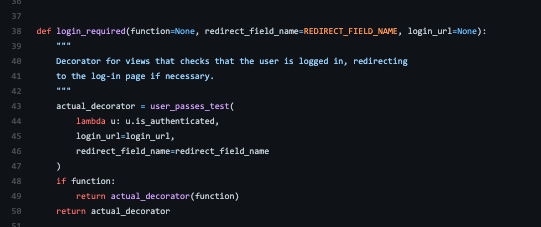
Github Library: <https://github.com/django/django/blob/main/django/contrib/auth/backends.py>

****

We are using this function to authenticate the user by giving it a username and a password. It will either return with an authenticated user or it will return None. The raw password is from the UserCreationForm which is a form that creates a user with the given username and password.

**django/contrib/auth/decorators/login\_required**

**Github Library:** [**https://github.com/django/django/blob/main/django/contrib/auth/decorators.py**](https://github.com/django/django/blob/main/django/contrib/auth/decorators.py)

****

This is used to call a decorator which is essentially something that will restrict what the user will be able to view on a page depending on a specific condition. Login\_required is going to see if the user is logged in and if the user is logged in then it will execute the view normally, but if the user isn’t logged in then it will redirect to a different path and the view will not be loaded.

**Django/contrib/auth/forms/UserCreationForm**

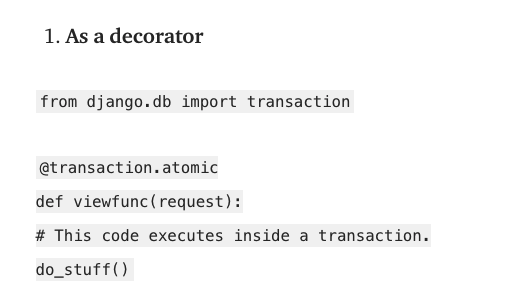
**Github Library:** [**https://github.com/django/django/blob/main/django/contrib/auth/forms.py**](https://github.com/django/django/blob/main/django/contrib/auth/forms.py)

UserCreationForm is a way for users to be able to sign up with their own inputs. It is a form that is used to create a user with no privileges with the username and password inputs that they give. We use this in our signup function which creates a form from the request with the username and password and then

saving it into our database.

**django/db/transaction**

**Github Library:** [**https://github.com/django/django/blob/main/django/db/transaction.py**](https://github.com/django/django/blob/main/django/db/transaction.py)

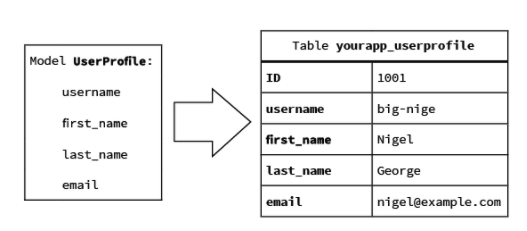
****

Django recognizes a transaction as an atomic set of database queries and will provide us with an API to control the transactions. This atomic property allows us to create a block of code that will check the transactions and if this block of code is cleared then the changes will be automatically saved to the database, if it’s not then there will be an error and the changes are rolled back. We are using this as a decorator to make sure that our user\_form and profile\_form is both valid when we have to update our profile, if either of them are not valid this user is not authenticated and so we would have the changes rolled back. Please I am so tired.

**More Information about Django’s Database**

Django provides a way for us to have a database through it’s ORM (object-relational mapping) which is connected to its underlying database, it allows us developers to work with data and the relational databases without having the complications of having to learn the database structure and takes away from us requiring to learn SQL to change and manipulate the data. Django provides us a way to only code in Python and it will convert all the code that we have into SQL language.

We use models in django to reference the data in the databases. When our models are created, Django will turn the code into SQL and will create a database corresponding to our customized model. This is an example of how the code will look after Django has changed it to SQL.



Once our code has been converted and we have created the tables in our database, we will use relationships to keep track of the data in these tables. These relationships between the model and the database are connected via a “foreign” key which is linked to the user\_ID. This makes it so that each user’s information has a unique foreign key and their table will only be linked to this key. It’s sort of like going to the bank and each bank account has a special pin and then inside you have multiple checking accounts.