**User registration 2**

From user registration part 1, when you click on click on submit the request will go to **register.html(based on form action).** If we go to **urls.py** we can see that whenever someone calls for register it is firing up the **views.register** which takes us to the views.py where we have the register function returns the **register.html**. we are using the same function for fetching the page and for sending data to table as well. That’s where the GET and POST makes sense. When you are calling the **register.html** you are sending a **GET request**. When you are **submitting the data** you are sending a **POST** request. So even if your URL is same what is changing is the type of request. At the moment we are submitting a GET request(return render(request , ‘request.html’))

What if it is a post request- we need to check if the request is GET or POST that’s where we use if else and we will map the variables to the html fields on the index.html page as shown below.

Graphical user interface, text

Description automatically generated

We got the data from user, how will we save this user data in the database, usually we need to write sql query. In this case, we are using ORM and once you got the data we can push it to database table provided you have a model with you. To recollect if we have a model object we can set this data there and we can save this data in database directly. Question now is do we have a model object for the user, in Django it is available within the framework and we just have to use it. How to use it? –

Import the auth.models and user, auth. Let’s save this data in object user =user.object(to handle the database).create\_user(create a new user) and pass all the parameters.

Save the data – user.save(), print something and return or redirect to home page

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Let’s verify if it works. Fill the form and submit the details and it will route to the home page.

Graphical user interface, text

Description automatically generated

Notice that our cmd/console shows user created.

Text

Description automatically generated

Let’s verify the database table. Auth\_user 🡪 right click 🡪 view all data. We should see the user got added to the table. Discuss all the fields, password is also stored in secure format.

Graphical user interface, application

Description automatically generated

We are still missing some things here:

1. We are not checking if the password1 is matching with password2 during the registration.(we need to have same password)
2. What if the username is already taken or used? – they should be unique. Think of gmail or some other provider saying this email or account id is already taken.
3. We are not checking if the email id is already exist in the database or not. If it exist we can’t use the same email ID again.

If password1 == password2:

#conditions here

Else:

Password not matching

Graphical user interface, text, application, email

Description automatically generated

Now if we submit the form and give a try registering with the unmatched passwords. We will receive the following message in the console. NOT The new user created message.

Text, letter

Description automatically generated

2nd checking we have to do is if the user is already exist in the database, even if it is same or password is same. Let’s first check for the user(user.objects.filter) and email ID check as well.(user.objects.filter(email….))

Application

Description automatically generated with low confidence

Let’s try registering again, click on register and fill random values – use the username which is already taken. It will give the username taken message in console, we can try it for password as well.

Text, letter

Description automatically generated

We have checked for the username, email and if everything is matching then it will save as a new user.

Next we will look at the login, logout and work with messages part. We are displaying the messages in the console, we want to display the message on the same page where we are performing operations.

Recap: we have seen how to register the user in the database