django report III

NOTE
This is django report III
If you want to see the licence or the overview of what django is please see django report

Database

Postgresql: This is the databased used to hold the information from this phase and django specifically supports it. Databases are formatted in tables with each column being a different data attribute. In the case of django each column of the database is a field of a model. For example a username of a user would be a model textfield and that field would be a column in a database table. Django is able to connect to the postgresql server and database and then login in order to store and get data. We used postgres to store all user information. To see how postgres is connected to by django:

https://github.com/django/django/blob/master/django/db/backends/postgresql/client.py

Models with manytomanyfield: Models was covered in django report 2 this will be on the manytomanyfield specifically. The manytomanyfield allows for one class to hold reference to another class which can be stored. This can either be done directly or through an intermediate class, which is better when a relation is 2 ways.

This feature was used to create the like functionality and the following functionality. Likes was done by adding a manytomanyfield to our post class and adding/removing user accounts when ever there is a like. A method called count can also be used to get the number of accounts which has liked the post.

Following was done having followers and followees in a follows intermediate class so each user has both a list of who is following them and who they are following. The relationships are not symmetric however are related and thus passed together.

https://github.com/django/django/blob/ccb1cfb64e919e163c51995ed99bff3c92d7d006/django/db/models/fields/related.py#L1120

Models with manytomanyfield: Same as manytomanyfield above however it is only used to assign one class instance to another. We used this to connect the users class from last phase to a new user profile class which holds the user account info.

https://github.com/django/django/blob/ccb1cfb64e919e163c51995ed99bff3c92d7d006/django/db/models/fields/related.py#L1120

Save: A part of how models communicate with the database this saves the changes made to a class and saves it to the database. This is used for when a change to something stored in the database is made. For example to change a user profile.

https://github.com/django/django/blob/ccb1cfb64e919e163c51995ed99bff3c92d7d006/django/db/models/base.py

JSONRESPONSE: This sends a http response and more specifically in the form of JSON. This is done to send data back to the path which made caused the update so that it can be dynamically updated. For example sending a jsonresponse of the new follower count of a user after a new follower follows them.

https://github.com/django/django/blob/ccb1cfb64e919e163c51995ed99bff3c92d7d006/django/http/response.py

Channels: Channels is django's way of making code live through use of web sockets. It is about to connect, disconnect, and send to other channels. This was used in the creation of the chat app for the website. Channels allows use to send the message the connected channels and we can connect channels based on if they follow the user. The channel connects users then after a web socket message is received it is can be processed.

https://github.com/django/channels/blob/master/channels/generic/websocket.py