Project Report – Django Configuration – Back End

**Introduction**

In this deliverable I create the back-end database structure for the previously designed front-end of the fictional IMPACT volunteer organization. The database is implemented in SQLite through the Django web framework. The work involved configuring a models.py file to contain the database structure, configuring the database through the python command line, and modifying the existing views.py, and models.py. I also created a new file in the project under the app directory, called forms.py, which is used in pulling data from the forms in the template to the database.

Of note is that the Django project itself is still called “deliv3”. I did not appreciate how often I would be referencing the name of the project when working with Django while working on my last deliverable. This deliverable is a natural extension of the last, and so I am working with the same project files, under the same name, “deliv3”, which has become completely useless in terms of identifying the purpose of the project. I have learned that in the future I should use a more descriptive, and general, name for a Django project.

The requirements included a login page with user authentication to bring a user whose information is on the database onto a logged-in version of the site. I regret to report that I had run out of time to implement this final requirement due to multiple jobs outside this course, and so this requirement is not included in this report and is not represented in the Django project.

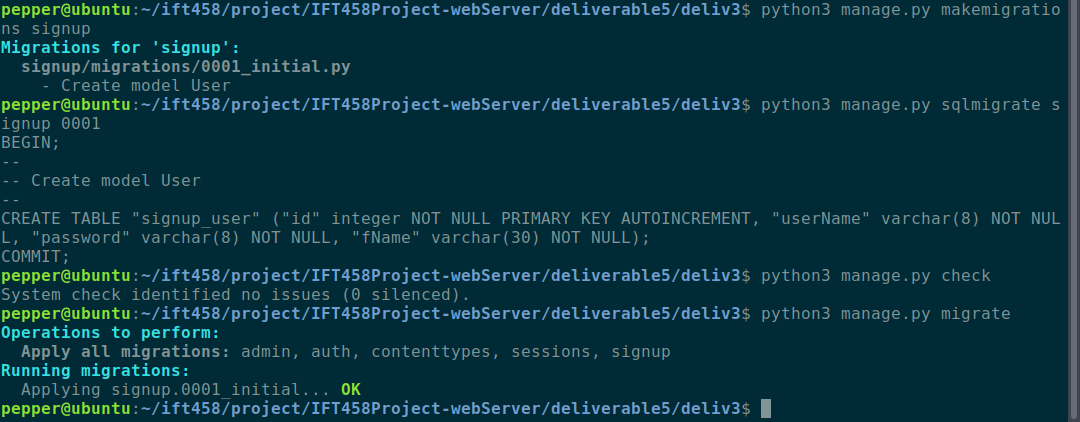
The missing requirement aside, I am proud that my Django project now allows a user to interact with the front-end in order to provide data that is stored successfully in the back end on a SQLite database, information which can be viewed and confirmed by a system administrator.

**Work Done**

***Configuring the Initial Database & Performing CRUD operations***

Work first began by making a simple database implementation in the newly created models.py file under the /signup/ application directory. This was done by following along Derek Banas’ excellent [YouTube tutorial](https://youtu.be/Wm8Eq0HIISA) on Django, which follows along with the official tutorial on the Django site. This video series was instrumental in my developing a base understanding of how Django works.

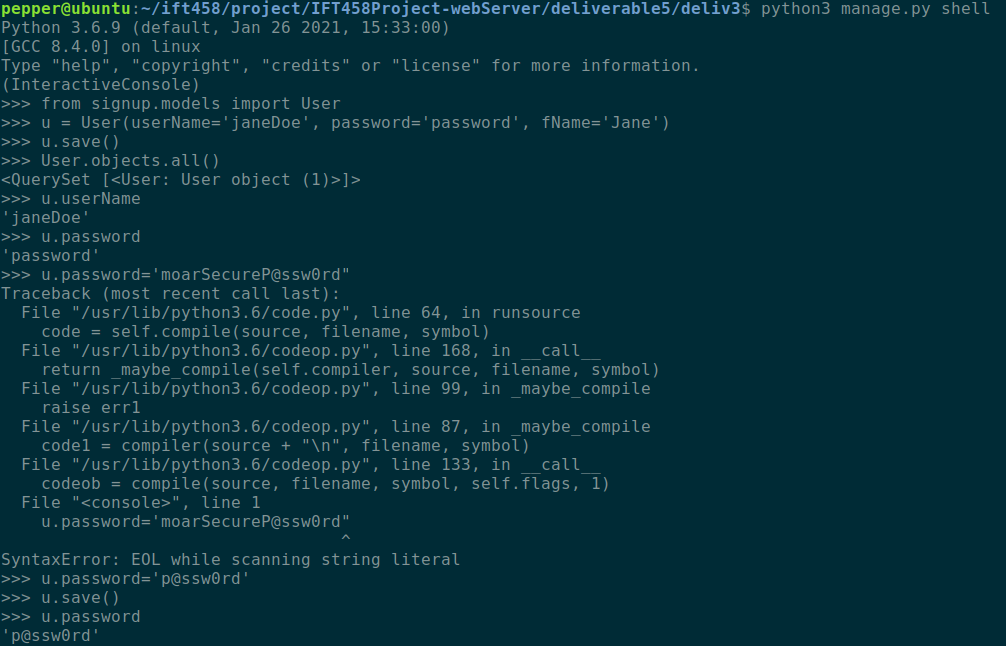
To make my basic database, I first created a class called User which will store all data entered on the signup page of the site. Within the field user I created three simple fields for the first three fields of my final design: username, password, and first name. Next I ran the `makemigrations` and `migrate` commands to create the database with these three fields under the column “User”:



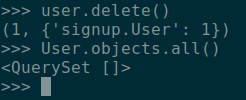
Creating the database using `python manage.py migrate`

Note that I had neglected to indicate the username as the primary key, and so Django automatically created one called “ID”. I later remedy this, and the primary key for the User models in the final implementation is the “username” field.

Next I test performing CRUD operations on this simple database. I do this in the python console, by importing the User model, creating a User entry, and modifying the data in that entry.



Creating, Retrieving, and Updating data



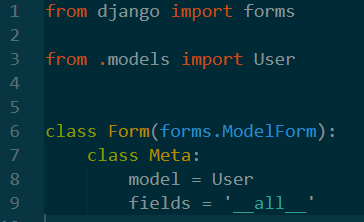
Deleting a column

You can see above that I successfully created an entry in the User column, with a username, password, and first name. Retrieving the entry’s fields is simple, and so is updating it. You can see that I call `User.objects.all()` to show how many entries exist, and after deleting the Jane Doe entry, making the same function call shows that the entry was deleted.

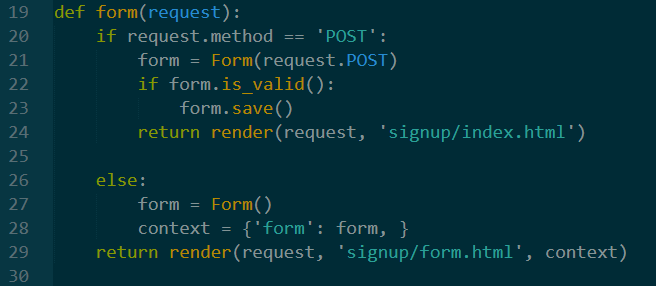
***Connecting the Database to the Templates***

In addition to the official Django documentation, I followed the “Django Model Forms” [tutorial](https://www.askpython.com/django/django-model-forms) on askpython.com to help me better understand how to implement my forms in my template within Django.

To facilitate handling the interaction between the fields and the forms in the template, I made a new file called forms.py:



I also created a new view to support the forms that posts the form’s data when submitted by the user:



To simplify the development process, I made a new template to display the new form to the user, called forms.html. It is important to note that signup.html is now no longer implemented in the project. The new template simply calls `{{forms.as\_p}}` which render the forms as a paragraph on the webpage. This implementation Is not as pretty as the original visual design of the signup.html file, but it is functional and simple, which meets the needs of the current task.

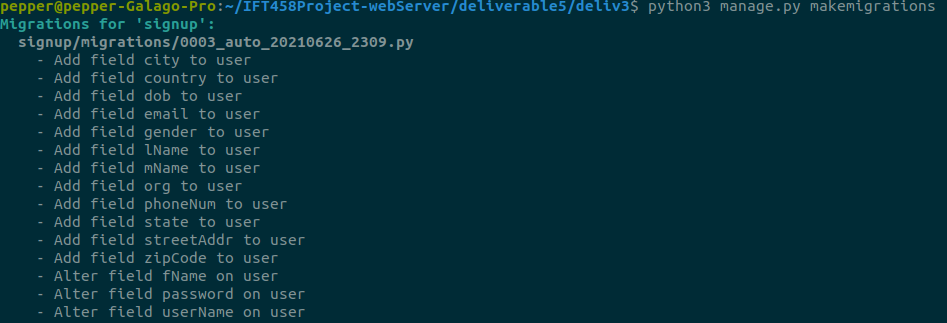
Finally, I updated the index.html file to point to forms.html instead of signup.html, updated the urls.py file under the signup app to include a url for the forms template, and adjusted the models.py file created earlier to include all the fields that the IMPACT signup page requires.

The project now allows you to run the server, open the site on the local host, navigate to the signup page, and enter information which is stored on the database.

**User Manual**

*Step 1:*

In the project root directory (the directory containing *manage.py*), open a command line and enter `python3 manage.py makemigrations`. If using Windows, instead of `python3` use simply `python`. Your output should be similar to below:



*Step 2:*

In the same directory, enter `python3 manage.py migrate`. The output should be similar to below:



*Step 3:*

Run the command `python3 manage.py runserver` to turn on the Django server.

*Step 4:*

In your web browser, navigate to localhost:8000/signup

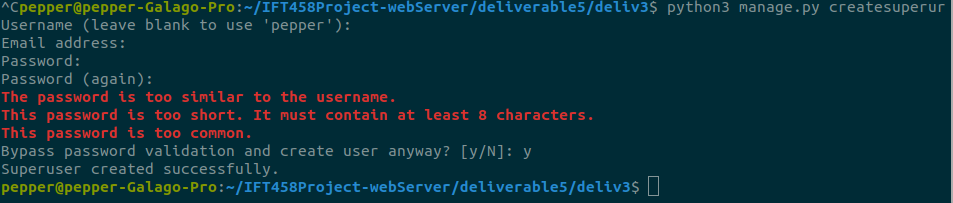
*Step 5:*

Navigate to “become a Volunteer” in the navigation bar, and enter information, overwriting the default values. Select the “Submit” button.

You will then be redirected to the index page, and your information is saved in the database. To verify this, create a superuser account on the server:

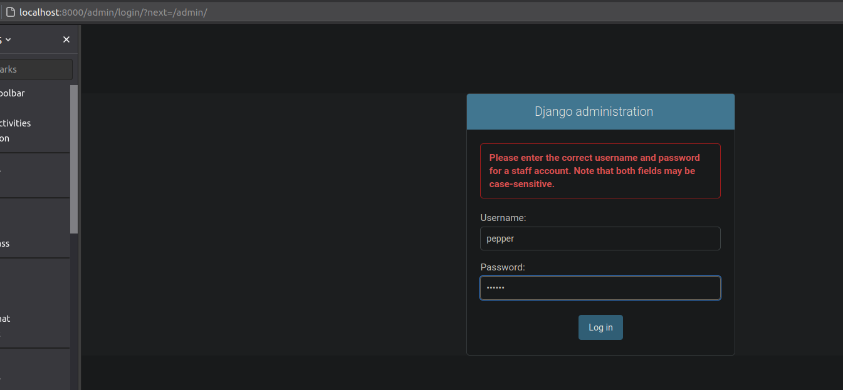
*Step 6:*

Run the command `python3 manage.py createsuperuser` and enter in your information



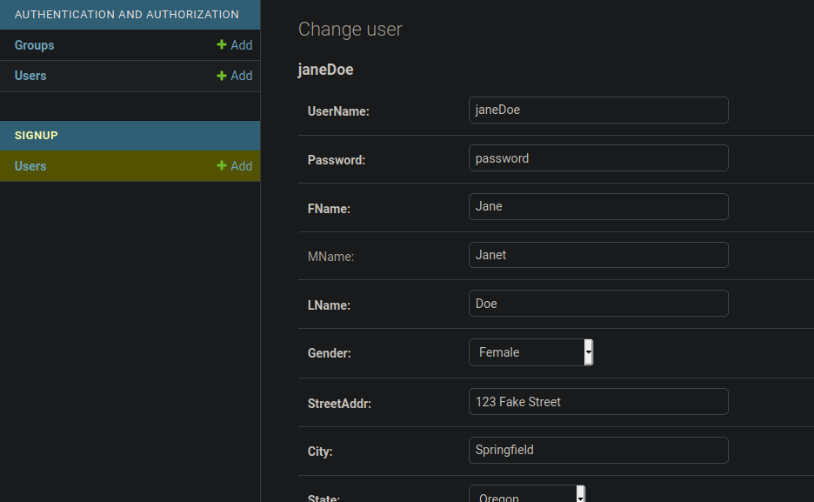
*Step 7:*

In the web browser, navigate to the address “localhost:8000/admin” and log on using your newly created credentials.



*Step 8:*

Click on the Users button and select the username of the information you had entered previously. You should now see the entry on the database along with the values of the fields you had entered:



**Conclusion**

In this deliverable I successfully connected a front-end design to a back-end database within the Django framework. I learned how models work in conjunction with a SQLite database and Django templates, and gained a deeper understanding of the function of views in a project.

I also gained a deeper appreciation for the complexity of Django that hides beneath the simple exterior. I remember reading in the lectures or elsewhere that being proficient with a framework requires deep understanding of how the framework works with consideration to each of its pieces. This became apparent in this deliverable, as I had assumed that this would be relatively straight forward, but I needed to learn a great deal about Django just to do this new simple task of pushing data to the database from a template.

I also learned a hard lesson in time management. I was not successful in completing all requirements of this deliverable, mostly due to poor time management and a poor work-life-school balance. In the future, I will appreciate that when working with and learning new software tools, I should give myself triple the amount of time I think I need to get the work done.

The challenges faced in this deliverable were similar to the last deliverable: learning Django has required me to re-shape how I think about software tools. I have so much experience with object-oriented programming that I became frustrated when Django did not behave as I expected it to in that lens. In having patience and allowing myself the time necessary to learn exactly where the logic flow is in configuring Django, I was able to overcome this challenge, to the degree that I could complete the majority of this deliverable on time.

My work can clearly be improved by including data validation on the forms and by making the forms.html template look more visually appealing. I would also like to go through the code and include more comments to document the logic flow of the app, so when I later return to this project as a refresher on Django, I can jump back into it easier.