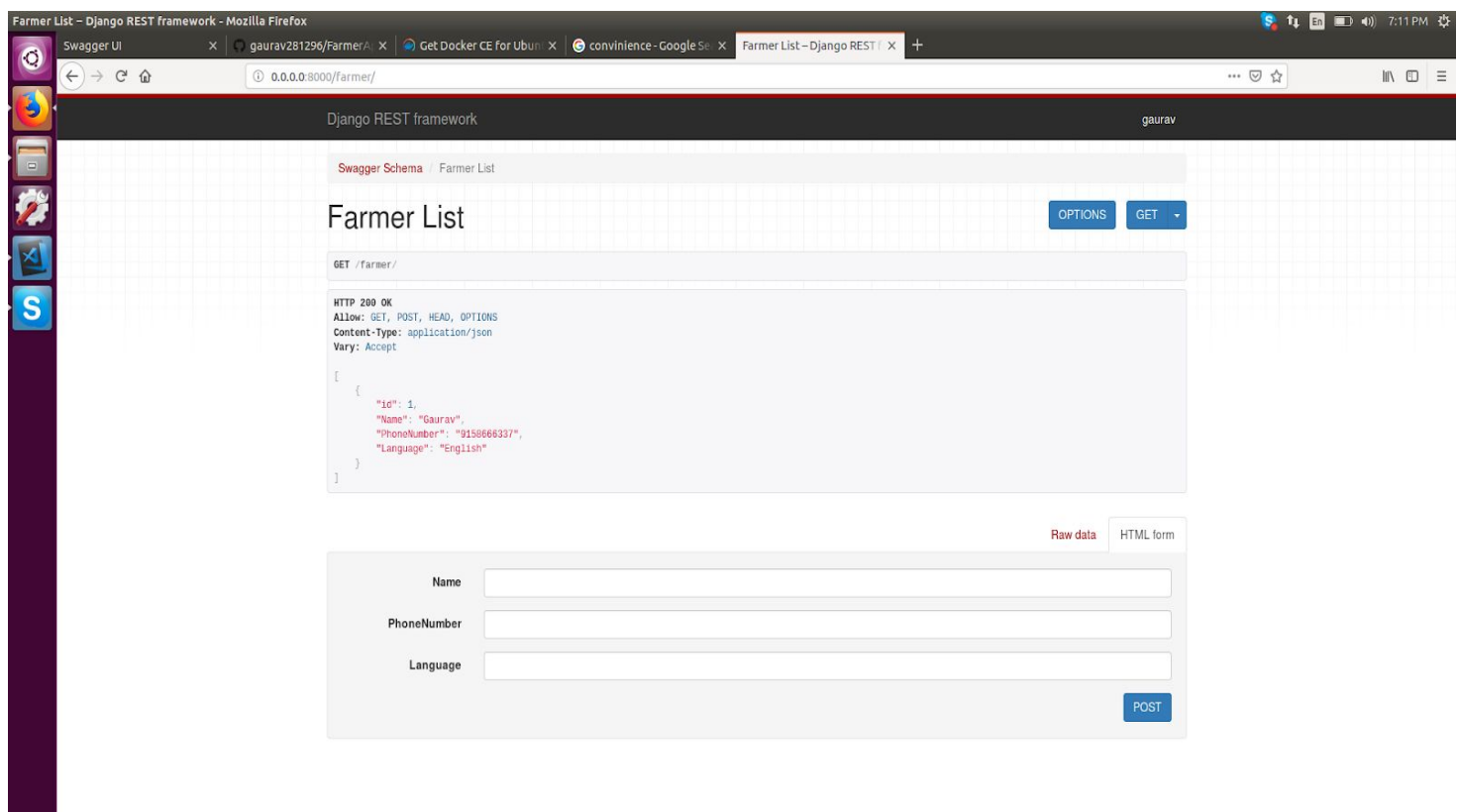


Farmer app for Shorelineiot

High Level Design

The application developed uses python3 with.djangorestframework. The entire application is highly decoupled in its design to maintain scalability and maintainability.

The Views which are returned to the user are made using generics and mixins that reduce the amount of code for performing the CRUD operations while at the sametime also provide us with browsable APIs such as the one presented below:



The business logic is implemented in the serializers.py modules for each app. This ensures us that any change in the model layer or the business logic would have no effect on the views layer. This is very import for maintainability of code.

The model layer interacts with the database. Have made use of sqlite3 as the database as it is provided out of box with.djangorestframework.

Functionality

The application provides the following features:

1. API authentication - By using the inbuilt api auth provided by django.
2. User management - By extending the inbuilt user to make a custom userprofile that has extra parameter - country. Used post_save signalling to maintain the User and userprofile table consistency.
3. Role Based Access Control - By making use of the inbuilt RBAC provided by Django - superuser, staffuser, user.
4. Swagger API help page - By using an external plugin that gives a view citing info about the designed APIs.
5. Browsable APIs - By making use of generics.
6. Deployments - Makes use of docker to run the server. The entire application is containerized and can be used inside a microservice platform such as Azure Service Fabric too.

Features that could not be delivered:

1. Switching of db based on user's country - I had never worked with Django before this and it took me a while to grasp the entire architecture of Django. I understood that you can switch the db by providing it's key from Databases in the settings.py file.

```
model.objects.using('country_db_key').all()
```

The feature is partially developed in dev2 branch. Where the farmer table properly switches the database as per the user context - user's country registered in the profile.

To make databases on runtime, I intended to make use of raw cursors to make new aliases for making new databases. Referring to this link:

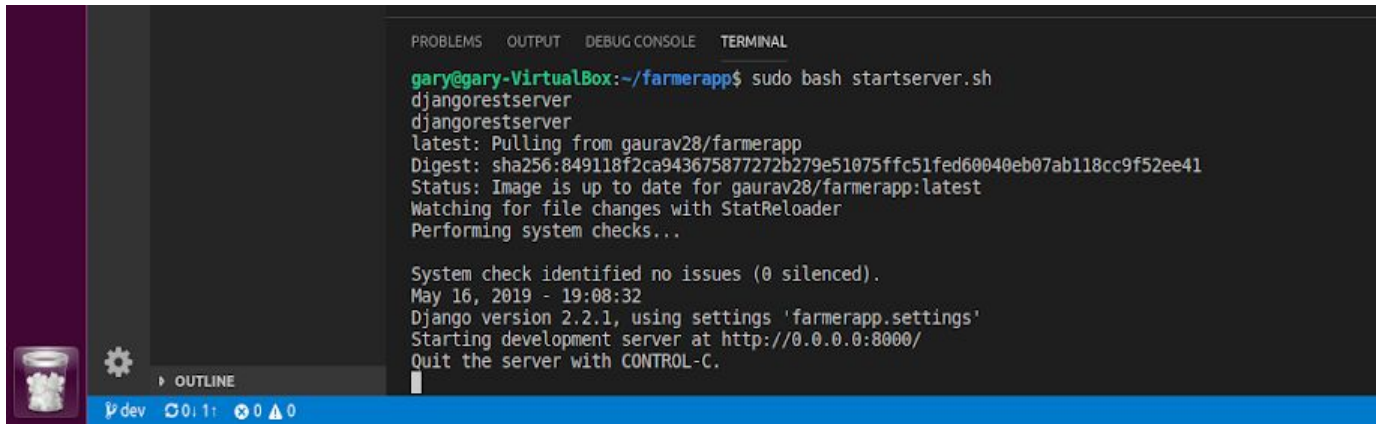
<https://docs.djangoproject.com/en/2.2/topics/db/multi-db/>

Unfortunately, I encountered some challenges while trying to have db switching on the basis of user context and had to revert my code to a state that was independent of the user's country.

2. Testing - I failed to provide test cases for 2 reasons. My lack of knowledge for testing and due to shortage of time to learn testing.

User manual

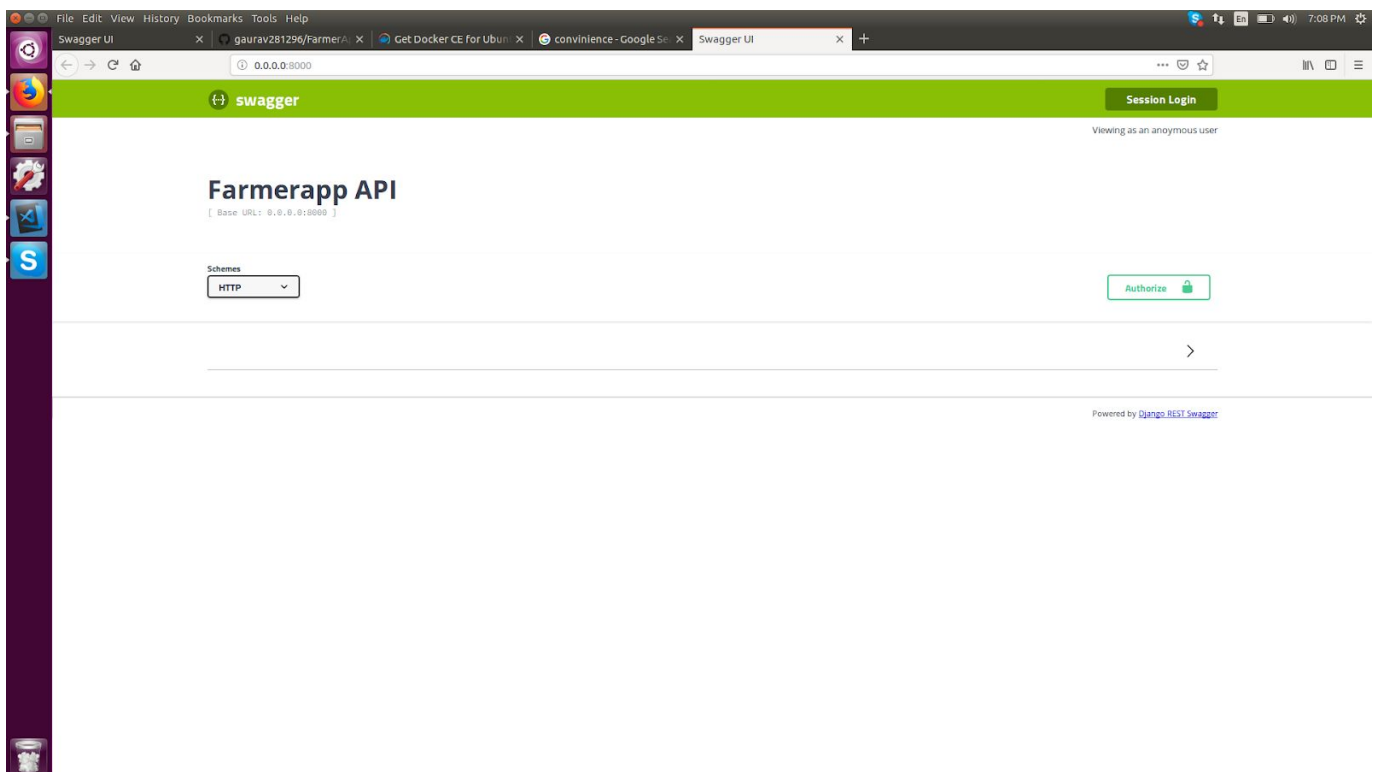
1. Once the setup is complete, start the server with the shell script provided. (It pulls a docker image for easy deployment.)



```
gary@gary-VirtualBox:~/farmerapp$ sudo bash startserver.sh
djangoestserver
djangoestserver
latest: Pulling from gaurav28/farmerapp
Digest: sha256:849118f2ca943675877272b279e51075ffc51fed60040eb07ab118cc9f52ee41
Status: Image is up to date for gaurav28/farmerapp:latest
Watching for file changes with StatReloader
Performing system checks...

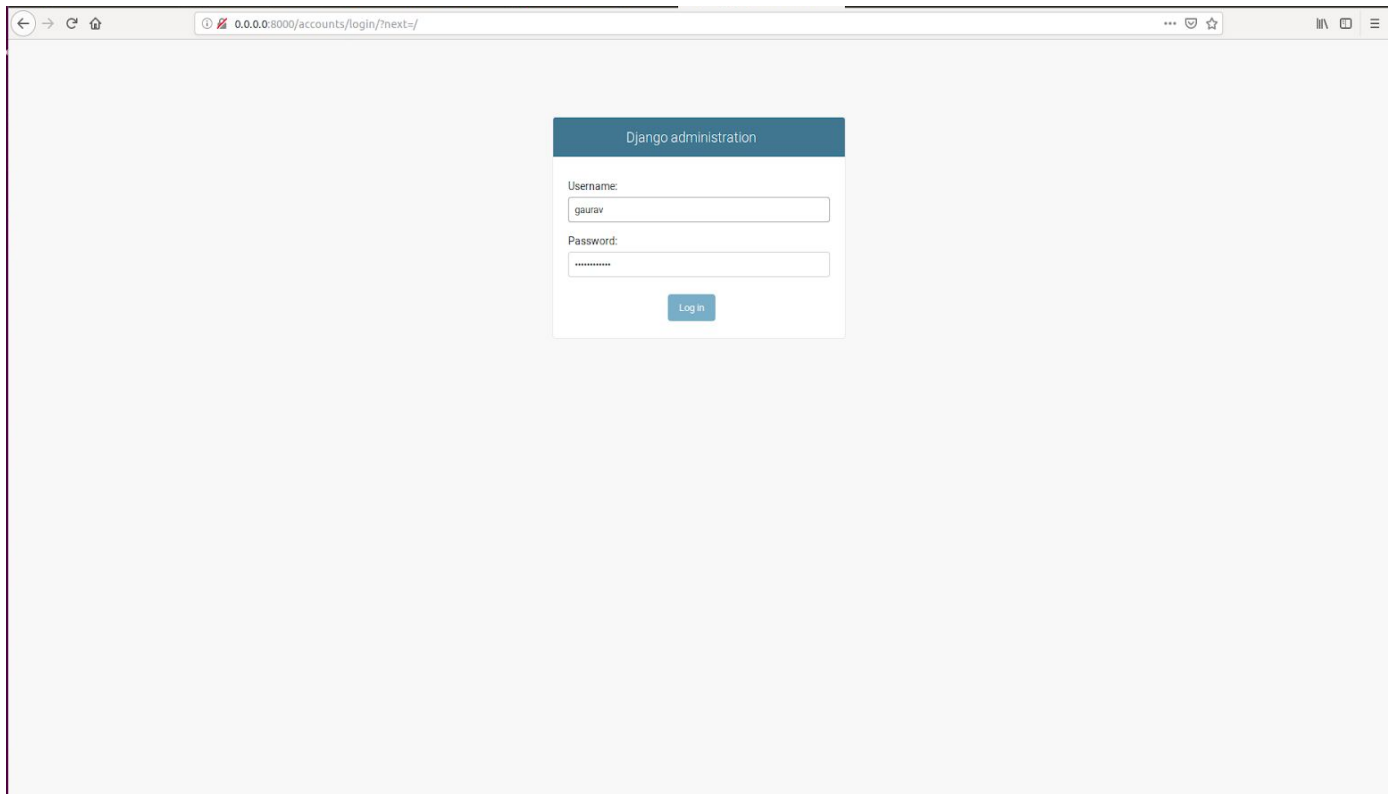
System check identified no issues (0 silenced).
May 16, 2019 - 19:08:32
Django version 2.2.1, using settings 'farmerapp.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

2. Log onto localhost:8000 to see the site page - the swagger doc:



You will not have access to the API help page until you login form the Session Login button.

3. Login to the app:

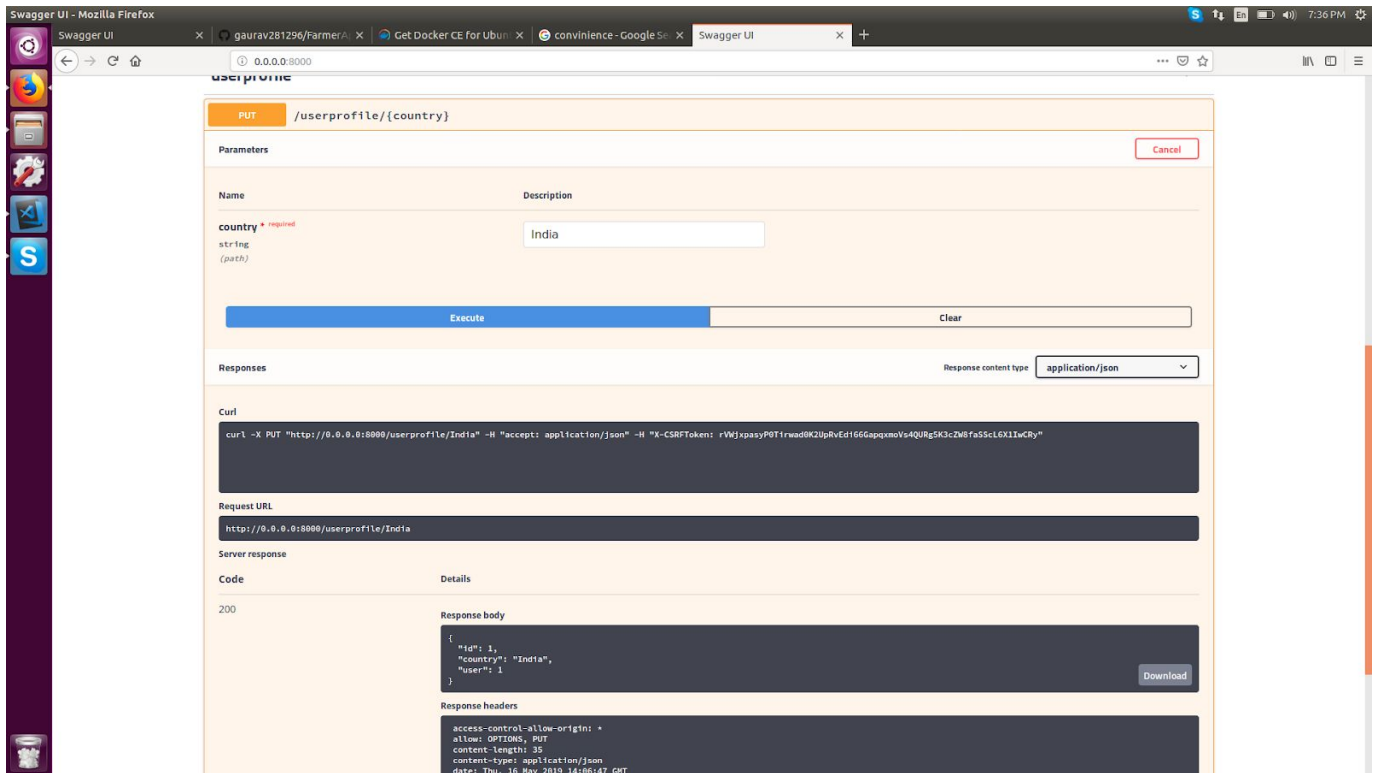


A super user has been created for you with the creds:

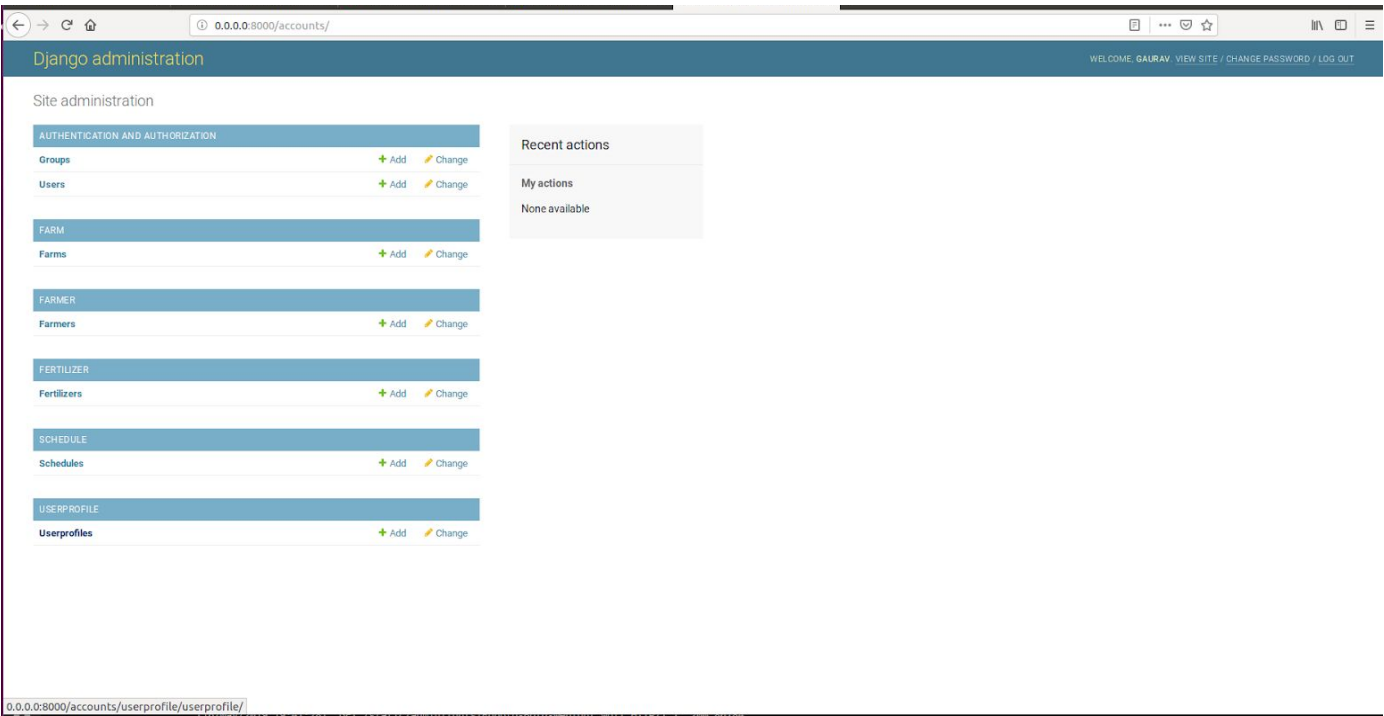
Username: gaurav

Password: shorelineiot

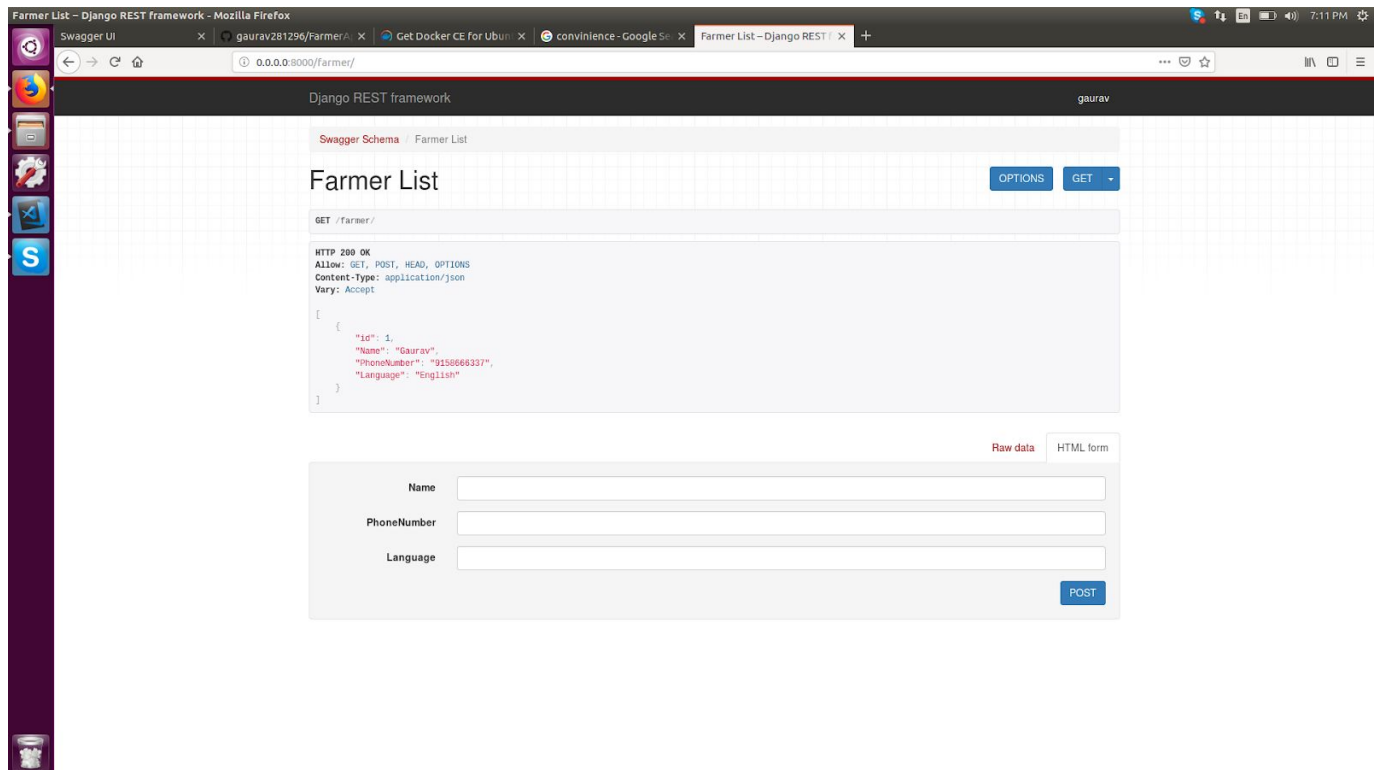
4. You can now access the swagger page and play around with it. You can modify the userprofile to set the user's country too. However, the user can only change his/her own country. Only the superuser has access to change other user's country via the accounts site (explained later).



5. The admin site is at localhost:8000/accounts
It can be accessed by staffusers only.



6. Since the APIs are browsable, you can visit their paths and use the django's generic view to query the data with a better experience.



Future scope

1. Implement db switch. I believe I was on the right track for achieving this. Need to look into better ways of implementing the same.
2. Implementation of test cases.
3. Currently the data in the container does not persist. This was done on purpose because the practice that I follow is to mount the data directory of the container to the /var/local directory in the host. This directory is usually empty until someone makes use of a Linux service that needs this directory in particular. Using /var/local ensures that a valid path (which definitely exists) will be provided to every machine where we wish to deploy our server.

```
docker run --name shoreline_db -e POSTGRES_PASSWORD=postgres -d -p 5432:5432 -v /var/local:/var/lib/postgresql/data/ postgres
```