**Firewall System**

**Project overview:**

Build a System / Network that will act as a Firewall to limit access of incoming requests from outside of System / Network to protect resources.

**Admin Role:**

Admin should be able to see the incoming requests and Allow / Deny them and also a notification alert in case IP address is not registered in database (neither allowed nor denied)

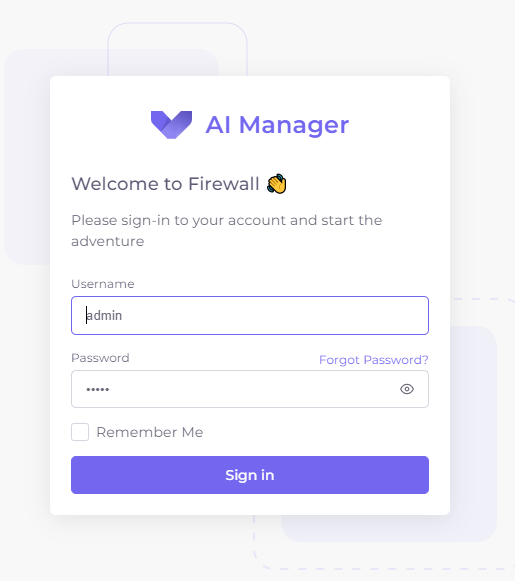
**System Resource to be Protected:**

Our system serves an API that returns a unique ID each time it is called. Firewall get triggered each time api is called.

**Working:**

As we know that our api returns a unique ID each time it is called and each time firewall get triggered.

1. System starts working by running at http:127.0.0.1:8000 or localhost:8000. We run the command python manage.py to start the system / server.
2. Admin has to login into the dashboard using admin and both username and password. This data about login is stored in Django’s built-in table called User.



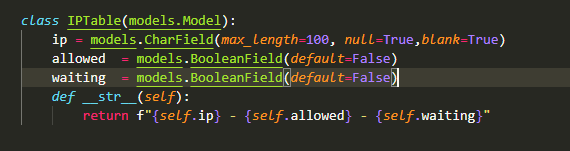
The form submit action will call a route in app > URLs.py which will further call a function loginRouter inside views.py and authentication would be done. After Login admin would be redirected to dashboard.

path('login/', loginRouter, *name*='loginRouter'),

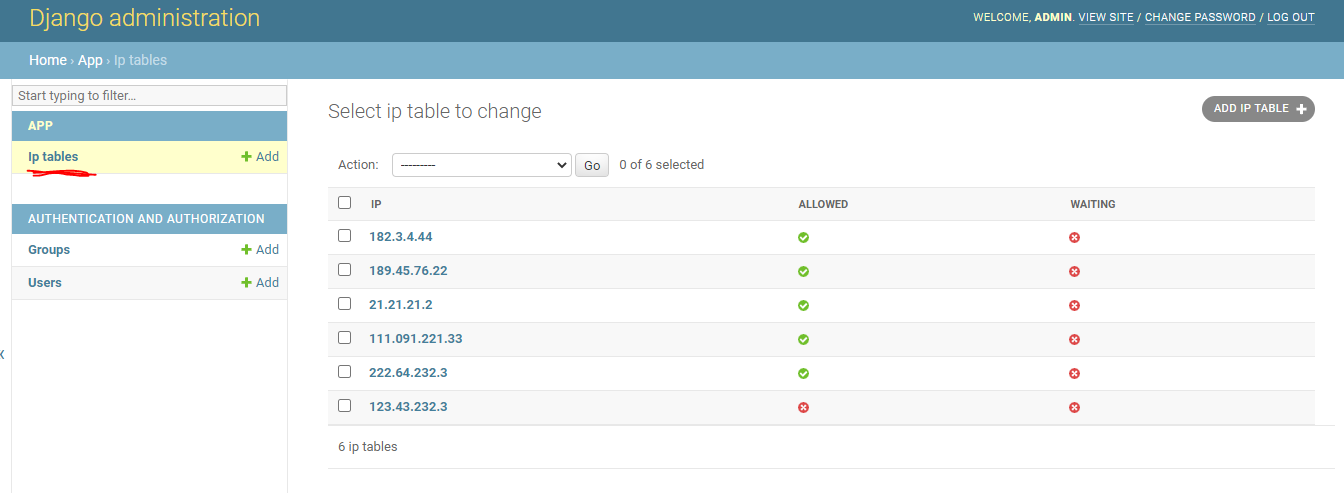
1. On dashboard admin can see a list of all registered IP addresses so far and change their status (Allow / Deny)
2. In models.py we have a model that act as a SQL table for keeping record of IP addresses.

An IP address can either have bool status for allowed (column) as True / False.

An IP address would have waiting set to True if its new inside system and admin has not made any decision so far about it (neither allowed nor denied)



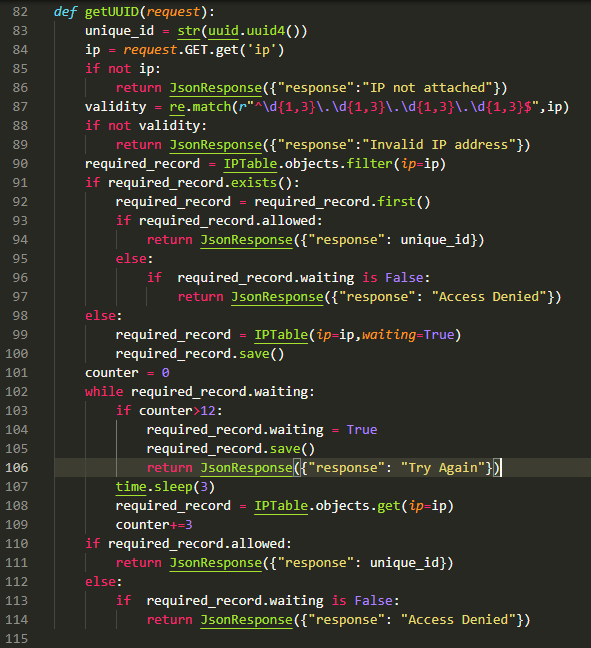
For this model we have following representation in django admin panel



1. The syntax for accessing our API is [http://localhost:8000/getUUID/?IP=IP-ADDRESS](http://localhost:8000/getUUID/?ip=IP-ADDRESS) this URL is handled by a route in app > URLs.py

    path('getUUID/', getUUID, *name*='getUUID'),

1. Inside getUUID function we extract IP address and perform following operations
   1. If IP is in invalid format system return error message
   2. If IP is allowed by system then return unique ID
   3. If IP is denied by system then return “Access denied” message.
   4. If IP is new to system (not in database) then we store it inside database with waiting status set to True
   5. After this the getUUID function waits for timespan of 1-12 seconds so that it may get marked (Allow / Deny by admin) and return response according to that. During 12 seconds if admin don’t mark it Allow / Deny a message returned by server saying “Try Again”.



1. On the end, on frontend, a setInterval function will wait of 1 second repeats a function inside it and that call an AJAX request to a URL getWaitingList

path('getWaitingList/', getWaitingList, *name*='getWaitingList'),



7.1 This function returns those IP addresses who are in waiting list with waiting status set to True inside database.

7.2 This function return IP addresses who are in waiting list and also plays an audio alert if any IP is in waiting list.



7.3 The ajax call on frontend receives a response and re-render the Table on frontend and shows the new IP addresses.

7.4 From there admin mark them Allow / Deny

7.5 If admin mark (Allow / Deny) new IP within 12 seconds after alert sound server will send response accordingly else now this IP has been registered in database and whenever new request would be made from this IP user will get response accordingly