# CORS, CSRF, and Session Configuration in Django
## (dj-rest-auth)

Steps followed to configure CORS in a Django project (for this project frontend is Angular)

1. Installed CORS package in the project with the command.
    - `pipenv install django-cors-headers`
2. Added `corsheader` in `INSTALLED_APPS`
    - ```
      INSTALLED_APPS = [
      ...,
      "corsheaders",
      ....,
      ]
      ```
3. Added `corsheaders.middleware.CorsMiddleware` in the middlewares list in `settings.py` just above the `django.middleware.common.CommonMiddleware`
    - ```
      MIDDLEWARE = [
      ...,
      "corsheaders.middleware.CorsMiddleware",
      "django.middleware.common.CommonMiddleware",
      …,
      ]
      ```
4. Added a list of allowed origins in `settings.py` to specify which origins' requests will be served.
    - ```
      CORS_ALLOWED_ORIGINS= [
      "http://localhost:4200",
      "http://127.0.0.1:4200",
      "http://localhost:80",
      "http://67.207.93.188:8081"
      ]
      ```
5. Set the following values in `settings.py` to configure CORS settings and Session and CSRF cookies.
    - `CORS_ALLOW_CREDENTIALS = True`
      So the browser can send cookies in its request to our domain, and subsequently, the browser can store cookies returned in the response.
    - `CORS_ORIGIN_ALLOW_ALL = False`
      Every origin shall not be entertained/served and if this is set to `True` then the list we defined in Step# 04 is unnecessary
      ```

- `SESSION_COOKIE_SAMESITE = 'None'`
  `CSRF_COOKIE_SAMESITE = 'None'`
  The browser will not check if a request is from the same site or not, this header prevents requests from malicious sites if the CSRF or session ID has been compromised

- `CSRF_COOKIE_HTTPONLY = False`
  `SESSION_COOKIE_HTTPONLY = True`
  If this is set to `True`, client-side JavaScript will not be able to access the Session and CSRF cookies. And if this is set to `False`, it's vice-versa.

- `CSRF_COOKIE_SECURE = False`
  `SESSION_COOKIE_SECURE = False`
  If this is set to `True`, the cookie will be marked as “secure”, which means browsers may ensure that the cookie is only sent under an HTTPS connection. For development settings, keep it `False`, so you may serve your application over HTTP instead of HTTPS. If you're in production, you should serve your website over HTTPS and enable `CSRF_COOKIE_SECURE` and `SESSION_COOKIE_SECURE`, which will only allow the cookies to be sent over HTTPS.