- **Retrofit** is a **REST** client for Android, Java and Kotlin developed by `Square`.
- The library provides a powerful framework for authenticating and **interacting** with APIs and **sending network requests.**
- This library makes **downloading JSON** or XML data from a **web API** fairly **straightforward**. Once the data is downloaded then it is **parsed** into a Plain Old Java Object (**POJO**) which must be defined for each "resource" in the **response**.

How to use :-

- Retrofit **turns your HTTP API** into a **Java interface**.

```
public interface MLModelService {
@POST("uploadfile")
@Multipart
Call<MLResponse> upload(@Part MultipartBody.Part filePart);
}
```

- The Retrofit class generates an implementation of the GitHubService interface.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl(https://qurany.herokuapp.com/")
    .build();
MLModelService service = retrofit.create(MLModelService.class);
```

- make a **call** and get **Response**

```
 repository.uploadFile(file).enqueue(new Callback<MLResponse>() {
        @Override
        public void onResponse(Call<MLResponse> call, Response<MLResponse>
response) {

        }

        @Override
        public void onFailure(Call<MLResponse> call, Throwable t) {
        }
    });
```

Our App

1. create interface, service, build retrofit.

- use `Multipart` to **upload** auido file as encoded in `POST` request.

2. prepare audio file.
   - make file object `File file = new File(path); // path of audio file`.
   - create `RequestBody` to be sent with `POST`

```
    RequestBody requestBody = RequestBody.create(MEDIA_TYPE_AUDIO, file);
    requestBody = new MultipartBody.Builder()
            .setType(MultipartBody.FORM)
            .addFormDataPart("file", "filename.mp4",
                    RequestBody.create(MEDIA_TYPE_AUDIO, fileObj)
            .build();
```

3. upload audio to api using **Retrofit**.

```
 repository.uploadFile(file).enqueue(new Callback<MLResponse>() {
        @Override
        public void onResponse(Call<MLResponse> call, Response<MLResponse>
response) {
                // get response as text of Recognized Text.
        }

        @Override
        public void onFailure(Call<MLResponse> call, Throwable t) {
        }
    });
```

4. process response and diplay results to user.