We used a tool provided by another developer as an interface to download all of the issues from the React.js GitHub repo. The tool was downloaded from the following repository:

https://github.com/gavinr/github-csv-tools

The output from this tool was a csv file with all issues from the React.js repository, and the code below processes the data to provide us with our 1000 sample dataset for the following labelling and Machine Learning analysis.

Initialize Spark

In [ ]:
```
import findspark
findspark.init()
findspark.find()

from pyspark.sql import SparkSession

import pyspark;
spark = SparkSession.builder.appName('sampler').getOrCreate();
```

Import the CSV file

In [ ]:
```
all_issues = (spark.read
  .format("csv")
  .option("header", "true")
  .option("inferSchema", "true")
  .option("multiline", "true")
  .option("quote", '"')
  .option("escape", "\\")
  .option("escape", '"')
  .load("issues.csv")
)
```

Below we prepare the dataframe for sampling. We made an initial filter with three parameters:

1. We will use closed issues only.
2. GitHub considers pull requests issues, so we have filtered out these as well.
3. Only use issues after 2018. React was in it's infancy, and we noticed the issues back then were mostly internal and very different than more modern issues.

In [ ]:
```
from pyspark.sql.functions import col, year

closed_issues = all_issues.where((col("`pull_request.url`").isNull())
                                 & (col("state") == "closed")
                                 & (year(col("created_at")) >= 2018)
                                 )

print("Total number of issues:", all_issues.count())
print("Number of filter issues:", closed_issues.count())
```

First, we will select only the columns we think might be relevant to help us with our manual

datapoint labeling (there are 111 cols in the original CSV).

We will also add a new column that includes the year and month of the comment. We will stratify our random sample by year-month of 'creation_date' to try to get representative sample across the entire lifespan of the project. Some time periods may have had more activity due to recent major releases, and activity has picked up with the popularity of the project. We want to capture it appropriately.

In [ ]:
```
from pyspark.sql.functions import col, concat, year, month

closed_issues = closed_issues.select("html_url",
        "number",
        "title",
        "labels",
        "state",
        "locked",
        "milestone",
        "comments",
        "created_at",
        "updated_at",
        "closed_at",
        "author_association",
        "state_reason",
        "body")
closed_issues = closed_issues.withColumn("time_period", concat(year(col("creat
```

Take a sample of 1000 points from the closed_issues dataframe, stratified based on time_period to ensure we are getting a representative sample.

In [ ]:
```
from pyspark.sql.functions import lit
# Determine the required fraction out of the total. We want 1000 samples, but
fraction = 1059 / closed_issues.count()
# Generate a fractions column for each distinct time_period in our df
fractions = closed_issues.select("time_period").distinct().withColumn("fractio
# Generate the samples stratified based on the "time_period"
samples = closed_issues.stat.sampleBy("time_period", fractions, seed=0)
#ensure we have an adequete number of samples, after possible rounding errors
samples.count()
```

Add a new column for each labeller to store their label in.

In [ ]:
```
samples = samples.withColumn("Graeme", lit(0)).withColumn("Steve", lit(0)).witl
```

Export the final list of samples to an excel file for manual labelling

In [ ]:
```
panda_df = samples.toPandas()
panda_df.to_excel("ENSF 612 - Label Samples.xlsx", index=False, engine="xlsxwr
```