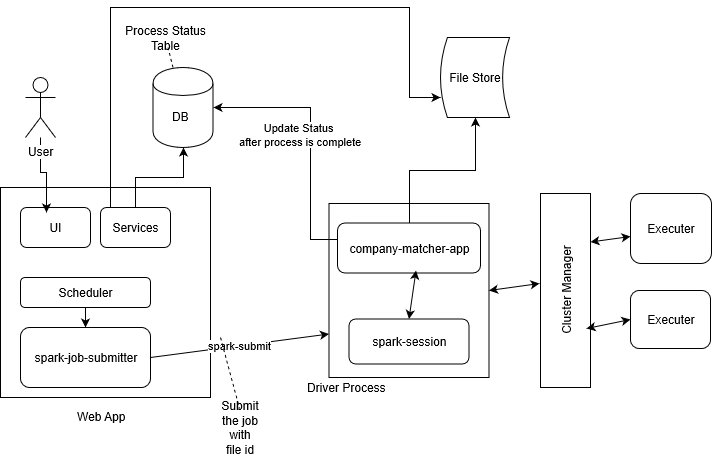
**Proposed Solution**:

1. User uploads the file. File id is generated and store in file storage (could be network drive, shared file system or s3 bucket). Request status is PENDING for newly submitted user requests (DB: REQUEST\_STATUS\_TABLE)
2. Job Scheduler which triggers in every 30 seconds (configurable time), picks up the PENDING requests from DB and submits to the spark (spark-job) programmatically. It also changes the request status from PENDING to PROCESSING
3. Once job is submitted to spark, company matcher app starts processing file. File id, location are provided as an argument to matcher-application. Once the process is complete matcher-application updates the DB to mark the request status as either PROCESSED or ERROR (REQUEST\_STATUS\_TABLE)
4. Request status UI shows the data from REQUEST\_STATUS\_TABLE.

Sample REQUEST\_STATUS\_TABLE Table data:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Request ID** | **Request Status** | **Request File  Content** | **User** | **Created On** | **Generated Data** | **Notification  Generated?** |
| 1 | Complete | [blob] | Riyaz S | 09-Feb-20 | [blob] | Y |
| 2 | Processing | [blob] | Riyaz s | 10-Feb-20 |  | N |



**Response to follow-up Questions:**

1. The companies.csv file we provided contains only 10K companies, but in reality we hold millions of companies in our DB. The matching method provided is intractable to be used on the full companies dataset. Can you describe a potential modification to the matching strategy to allow it to scale? (Hint: You don’t need to be comparing every uploaded record to every company)

My comments:

We can think of catching and avoid comparing every uploaded record to every company. Once matching record is found based on name and threshold criteria, we can cache this result in memory where key could name name-threshold -> ScoredMatch

1. How would you evolve your solution to scale to large upload datasets, for example 1M records?

My comments:

Apache Spark is a fast and general engine for large-scale data processing. We can leverage the features provided by spark such as data partitioning and create the partitions by “reduceByKey” or “groupByKey” to overcome any issue related speed of processing of 1M records.

1. Can you think of a way where you could notify users once a match job has been completed? How would you go about implementing this?

My comments:

We can configure the job which will send notifications after process is complete. We can do this for all the request with flag PROCESS\_STATUS = ‘COMPLETE’ and NOTIFICATION\_GENERATE = ‘N’)

**Feedback**

The test was good. I did not fine very difficult but it was it was lengthy to implement all the request features in given timeframe. I have spent around 10 hours of my time to solve the problem, think of proposed solution and prepare the required documents (including readme). Overall it was good learning experience for me not just as a developer but also as a technical architect ☺