|  |  |
| --- | --- |
| **Challenges** | **MAKS Approach** |
| Finding actual API URL of Bank sites | We used **Postman(A Chrome Add On)** to get actual URL |
| Website Design Change | We have Auto mailer program which triggers email in failed scenarios |
| PDF Data Extraction | We use In-House preparatory tool called **BEAT PDF Extractor** |
| Modification in Requirement - For e.g. Ticker Addition | It took less effort to modify as per new requirement due to generalization approach of the code |

Architecture:

1. Data Aggregation:

We explored different open source and preparatory tools. But they were not generic.

So we started exploring solution with Python, Java, C#.

We end up choosing Python and C# for following advantages:

* Python has enrich library pandas for capturing table data directly from the web URL
* Libraries like Selenium, Scrapy, Beautiful Soup are pretty straight forward and less time consuming for resources to complete scrapping
* Complex websites are easily doable with less development time in C# although the run time is more.

So, we did the tradeoff between Python and C# and used both as per the best scenario.

1. Data Engineering:

We performed following engineering on the data to ensure we have quality data in the data lake.

* Data duplicity Check at record level
* Introduction of Product code – A single attribute which provides information of all the important attributes to group the collected data.
* Data Formatting
* Data Filtering
* Data Transformation using Pandas

1. Data Analysis:

We have started following analysis on the collected data to get the insight with the help of our Domain Experts.

* Descriptive Analysis using **PANDAS**
* **Time Series** Analysis

1. Data Visualization:

We have started visualizing the collected data using following technologies to get more stories out of it.

* **BOKEH** Library of Python
* **TABLEAU**