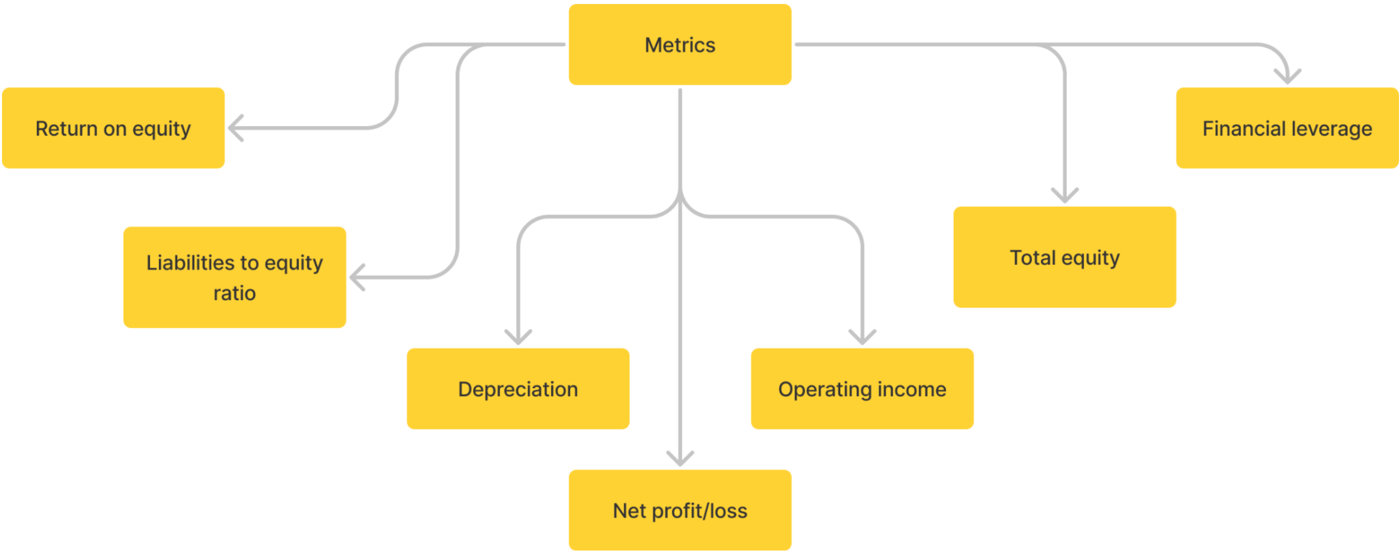
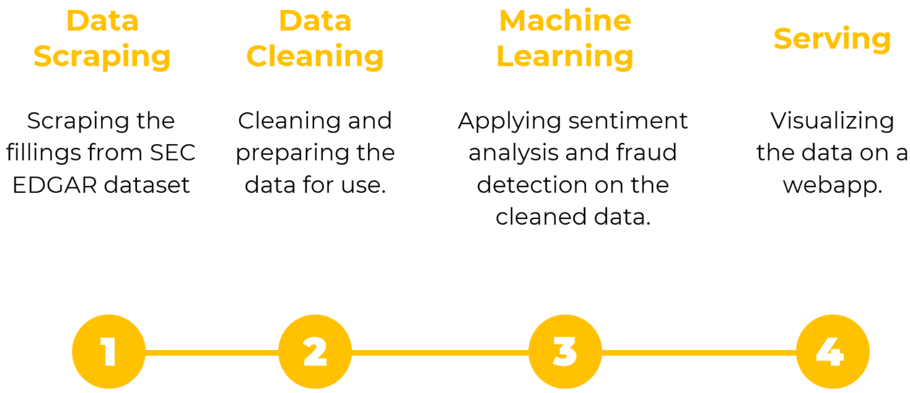
Digital Alpha's SEC filling Analyzer for SaaS Companies

Team 16

- **Abstract**
  - EDGAR, the Electronic Data Gathering Analysis, and Retrieval system, performs automated collection, validation, indexing, acceptance, and forwarding of submissions by companies and others who are required by law to file forms with the U.S. Securities and Exchange Commission (SEC).
- **Metrics showed**



- **Our Approach**



Data Scraping

Scraping the fillings from SEC EDGAR dataset

Data Cleaning

Cleaning and preparing the data for use.

Machine Learning

Applying sentiment analysis and fraud detection on the cleaned data.

Serving

Visualizing the data on a webapp.

Web Scraping

- **Abstract**
  - The primary task was to scrape 10-K, 10-Q, and 8-K forms. 10-K and 10-Q contain financial data (numerical values) however 8-K only contains an announcement. Since a pre-trained model for

sentiment analysis was found for raw 8-K filings, the task was reduced to scraping only 10-K and 10-Q forms.

10-K was the most important among all of these and contained the maximum number of values (300+ to be precise). No proper documentation was available to understand how filing works, grasping the structure required continuous experimentation and trial-error

- **Early Experimentations**
  - It was extremely challenging to understand the structure. Earlier the plan was to scrape the HTML document (which is in fact iXBRL). It contained lots of text which was not really required and many financial statements in the form of tables. These tables were supposed to be scraped but generalizing everything so that it works for all the tables across all the companies was not possible.

    After failing to scrape directly from the HTML, I experimented with other files available for each filing. Also, I noticed the button titled "Interactive" and it redirected to a page where the tables were displayed separately. Also, the fields present in the tables were interactive i.e. clicking on them revealed a box that contained "Definition", "References" and "Details". The section on details appeared important because it contained the name of the property (for example - us-gaap_ShortTermInvestments). At first look, this name looked like the primary key of all attributes in the table.

    Upon removing the last part of the URL (of 10-K filings) as shown below and checking the link, a list of files was found.

    https://www.sec.gov/Archives/edgar/data/796343/000079634322000032/0000796343-22-000032-index.htm

    This list mainly contained some XML files and some HTML files. The files of the form "R{number}.htm" were tables. But again scraping these were challenging.

    Every 10-K filing contained some data files which XML documents, these were as follows:-

    1. XBRL TAXONOMY EXTENSION SCHEMA DOCUMENT
    2. XBRL TAXONOMY EXTENSION DEFINITION LINKBASE DOCUMENT
    3. XBRL TAXONOMY EXTENSION CALCULATION LINKBASE DOCUMENT
    4. XBRL TAXONOMY EXTENSION LABEL LINKBASE DOCUMENT
    5. XBRL TAXONOMY EXTENSION PRESENTATION LINKBASE DOCUMENT
    6. EXTRACTED XBRL INSTANCE DOCUMENT

    These files contained all the data and the tables were probably displayed by extracting data from these XML documents. These XML documents contained all the information we needed to proceed but then again understanding these XML documents and scraping the useful data was a challenge.

- **Understanding the Structure**

  - After carefully observing the contents of these XML files, some dots got connected and the structure became somewhat clear. Three out of these files were most important for the scraping task. These were the Calculations file, Labels file and Extracted XBRL Document.

    The calculations file contained tags called “calculation link” which were basically identifying the table names (they were URLs but the last part was the name of the financial statement). And under every calculation link, locators were present. These locators were the attributes present in the table.

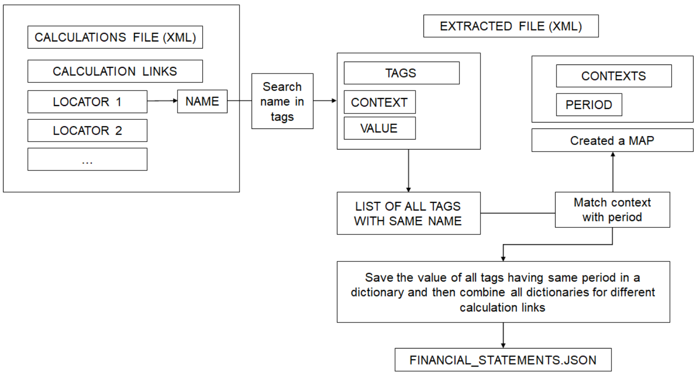
    The labels file contained all the attributes present in the form. These labels could then be stored along with their values to complete the scraping and make the database. But neither the calculations file nor the labels file contained any values.

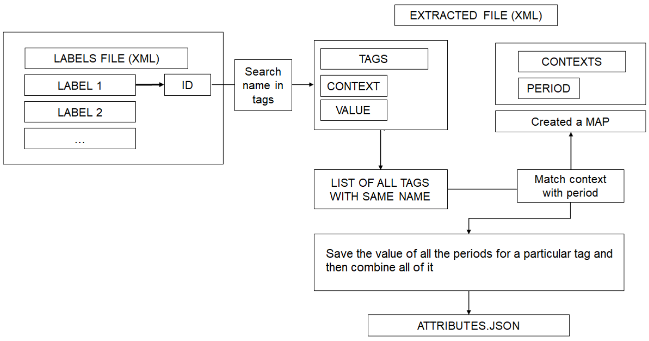
    The values were present in the Extracted XBRL Document. At first, it appeared easy to simply map the attributes found in the calculations and labels file to the values but the problem was extracted document contained many values for the same attribute. Structurally, every attribute was a tag in the extracted XBRL and the text in the tag was the value of that attribute. But the XBRL had many tags with the same name but different values. This was confusing but became clear later because the tables contained values for multiple periods and also, some attributes were present in more than one table. After lots of experimentation, I came across the fact that these different tags with the same name contain an attribute called context reference. The name of the attribute along with the context reference appeared like the primary key for all the attributes.

- **Scraping from XML data files**
  - Scraping the attributes out of the Calculations file and Labels file was easy. The challenging task was to find the values corresponding to each attribute in the extracted file. Firstly, I made a list of all the context (the context was of the form “{hexadecimal-number}_period”). Then, for each attribute, I listed all the tags which had the same name as the attribute. Obviously, these tags had different contexts and values, so extracted the period out of their context and saved their values accordingly into a python dictionary.

    The structure of the dictionary containing all the attributes was as follows:-

```
{

          ‘attribute’: {

                              ‘period1’ : ‘value’

                              ‘period2’: ‘value’

                              . . .

                    }

                    . . .

}
```


CALCULATIONS FILE (XML)
CALCULATION LINKS
LOCATOR 1
NAME
LOCATOR 2
...
Search name in tags
EXTRACTED FILE (XML)
TAGS
CONTEXT
VALUE
CONTEXTS
PERIOD
Created a MAP
LIST OF ALL TAGS WITH SAME NAME
Match context with period
Save the value of all tags having same period in a dictionary and then combine all dictionaries for different calculation links
FINANCIAL_STATEMENTS.JSON


EXTRACTED FILE (XML)
LABELS FILE (XML)
LABEL 1
ID
LABEL 2
...
Search name in tags
TAGS
CONTEXT
VALUE
CONTEXTS
PERIOD
Created a MAP
LIST OF ALL TAGS WITH SAME NAME
Match context with period
Save the value of all the periods for a particular tag and then combine all of it
ATTRIBUTES.JSON

The structure of the financial statements dictionary was also similar, the difference was that its key was the name of financial statements and for every table, it contained a set of periods and under the periods it contained a set of “attribute : value” pairs. The dictionary was then saved as a JSON file.

The period can be either an instance or a duration, it is denoted by the first letter of the key. If it is ‘I’ then it will be followed by a single date otherwise if it is ‘D’ then it will be followed by two dates corresponding to start and end.

- **Downloading the files**

For scraping all necessary information, 3 data files for each form filing was required. These were supposed to be downloaded and scraped.

- **Major Issue**

The first scraping script turned out to be very robust and worked for all the companies during implementation and testing, however, the fact that I only used filings of big organizations was a problem. Different companies organize their filings differently, the structure of XML they use are different, the context reference they use vary a lot, the attributes of tags they use may also vary and there are many more things that are not common among the filings of different companies. Also, the filings were different a few years ago and this also posed a problem because companies that did not have the latest filing could not be scrapped. Due to this, the first script was only able to scrape a little over 100 companies.

After spending a lot more time understanding the filings of different companies, I came up with another script that was able to scrape another 35 companies. In this script, a list of context references was made and a corresponding list of periods was made, and instead of extracting period from the context reference ID, it was replaced using the above map.

- **Extending to 10-Q**

  10-Q had a very similar format to 10-K. However, some attributes were not present in 10-Q and downloading 10-Q was more tedious than 10-K because it required scraping data from more than 1 filing (8 quarters to be exact). Except for a few changes, more or less the same script worked for 10-Q as well with a few tweaks and the data of 8 quarters was saved in JSON format.

- **Scraping 8-K**

  This was a relatively easy task because a model was found that can analyse directly with the raw 8-k filing data. So, the only requirement was to download the complete submission text file available on the filing web page. One major issue during downloading 8-K was that due to some untraceable reasons, the python kernel used to die randomly. After spending hours figuring out the problem and failing every time, I finally did the job manually by restarting the kernel every time it died and executing everything again from where it stopped the last time.

Machine Learning

Implemented Modules:

1. Fraud Detection Using Random Forest
2. Sentiment Analysis Using (Loughran-McDonald master dictionary)

Fraud Detection Module:

This model is used to predict if a given organisation is committing fraud. This is detected by evaluating the metrics of the organisation by a random forest classifier. Initially, we found 42 metrics that can be used for fraud detection (Appendix I), but because of the inconsistent metric name provided in the Edgar database we narrowed down the number of metrics to 13 (these were the most common present in the forms).

The data that we used to train our model was a compiled version of the AAER Database (Appendix I) consisting of the 42 metrics and the fraud label.

AAER Data: our initial accounting fraud sample comes from the SEC’s Accounting and Auditing Enforcement Releases (AAERs), as compiled by the University of California-Berkeley’s Centre for Financial Reporting and Management (CFRM).

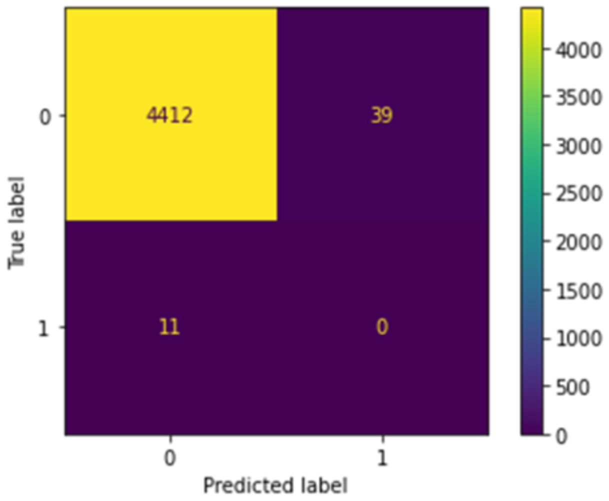
The AAER Data includes all the AAERs announced over the period between May 17th, 1982 and September 30th, 2016.

As mentioned in the research paper (Appendix I) the data in a range of +/- 4 years must be used for fraud prediction for a given year, so we dropped down our data to the range in 2012-2016. Further, some preprocessing was done to data to make it compatible with the current period.

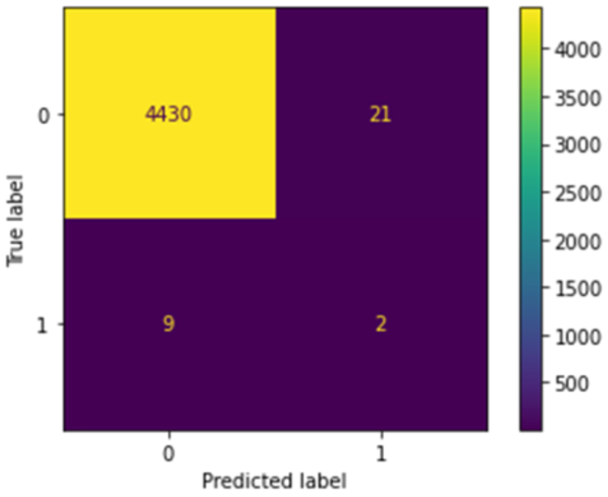
The data was highly imbalanced (20K –ve samples to 55 +ve samples), So we applied SMOTE(Random Undersampling and Oversampling) to handle the imbalanced classification. Furthermore, the parameters of the model were experimented with to handle the imbalanced classification. Our model achieved a ROC AUC score of {} during the testing.

To keep the codebase lightweight and computationally efficient the whole module was shifted to Google Cloud. Currently, we have five different versions of the model running on the cloud. The prediction from any of the selected models can be made easily and efficiently using the Google Cloud API. The default version for prediction is version 3.

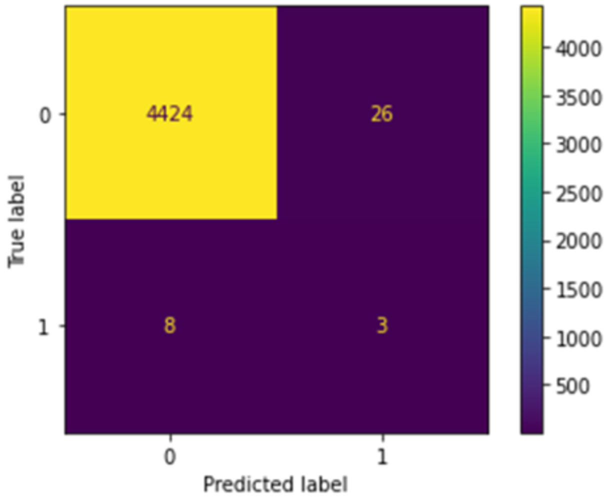
Model training performance:



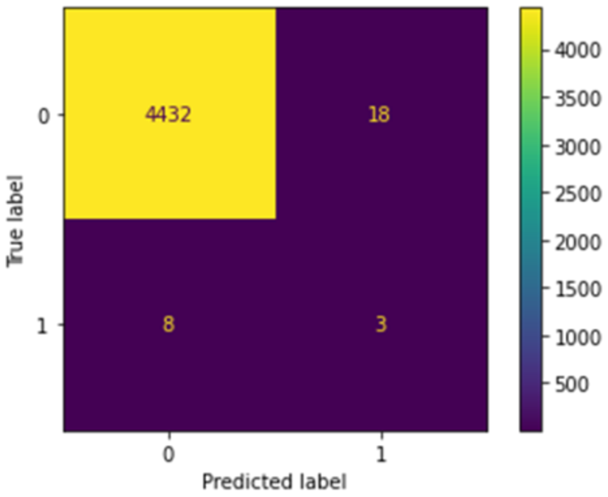
Neural Network



Random Forest



XGBoost



XGBoost with smote

Sentiment Analysis Module:

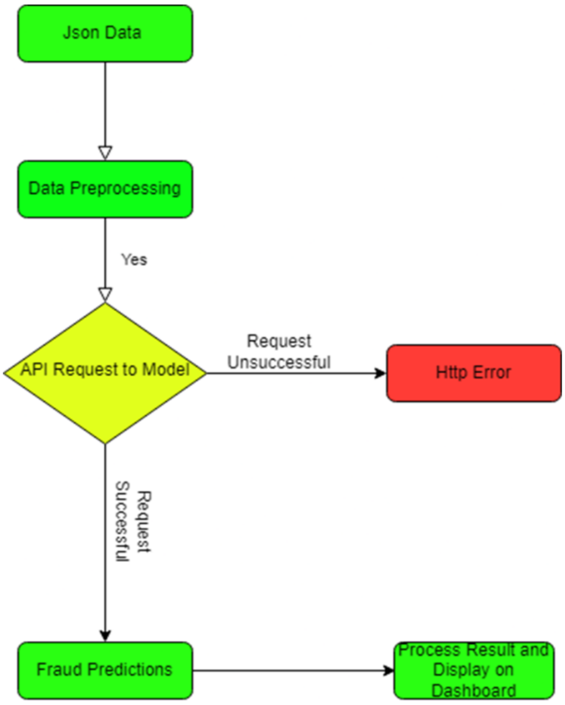
This module is used to conduct sentiment analysis on the given form (10-K or 8-K). This module uses the "Loughran-McDonald Master" dictionary (Appendix II) to provide a word-by-word analysis of the text extracted from the given form.

Loughran-McDonald Master Dictionary: The dictionary provides a means of determining which tokens (collections of characters) are actual words, which is important for consistency in word counts. Within the dictionary spreadsheet, we also provide flags for the sentiment dictionaries used in the JF paper (e.g., negative, uncertainty, litigious). Additional documentation for each word is also provided such as other classifications (e.g., Harvard TagNeg, constraining), syllabification.

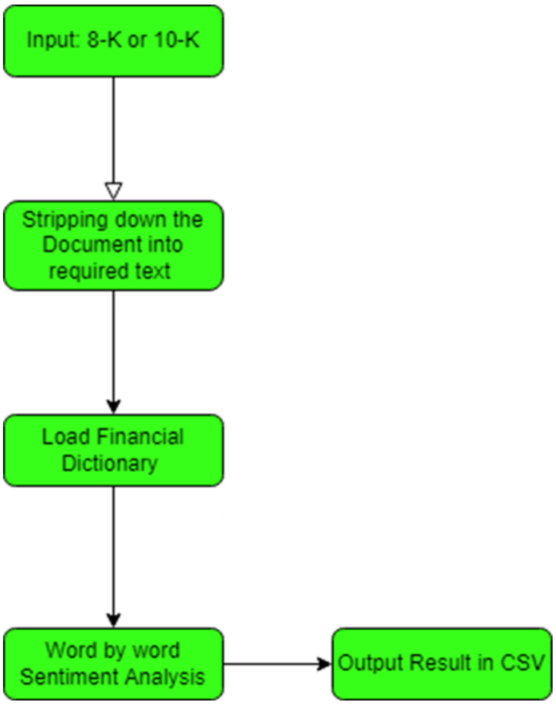
The text extraction is carried on by using the library Beautiful Soup(bs4). Various regex expressions are used to extract the required text from the documents. Once the extracted text is obtained it is passed on to a python script (Generic_Parser.py) which performs a work by work sentiment analysis on the file and provides a description of the whole text file in form of 6 6 variables (% negative, %positive, %uncertain, %litigious, %string modal, %constraining).

This provides a quick analysis of the forms, in 8-K as the major text contains announcements the analysis can be used to see if the announcements are positive (indicating company growth) or negative.

Model workflow:



Fraud Detection


Input: 8-K or 10-K
Stripping down the Document into required text
Load Financial Dictionary
Word by word Sentiment Analysis
Output Result in CSV

Sentiment Analyis

Why is the model never outdating?

- The models presented above can be easily retrained with new data and hence can be updated to a newer and better version.
- The model retraining process is simple, as the models are hosted on G-Cloud running a single script with the new data will automatically train the model and it will be ready for predictions

Appendix

I: Yang Bao, Bin Ke, Bin Li, Julia Yu, and Jie Zhang (2020). **Detecting Accounting Fraud in Publicly Traded U.S. Firms Using a Machine Learning Approach.** Journal of Accounting Research, 58 (1): 199-235.

II: https://drive.google.com/file/d/17CmUZM9hGUdGYjCXcjQLyybjTrcjrhik/view?usp=sharing

Backend Development

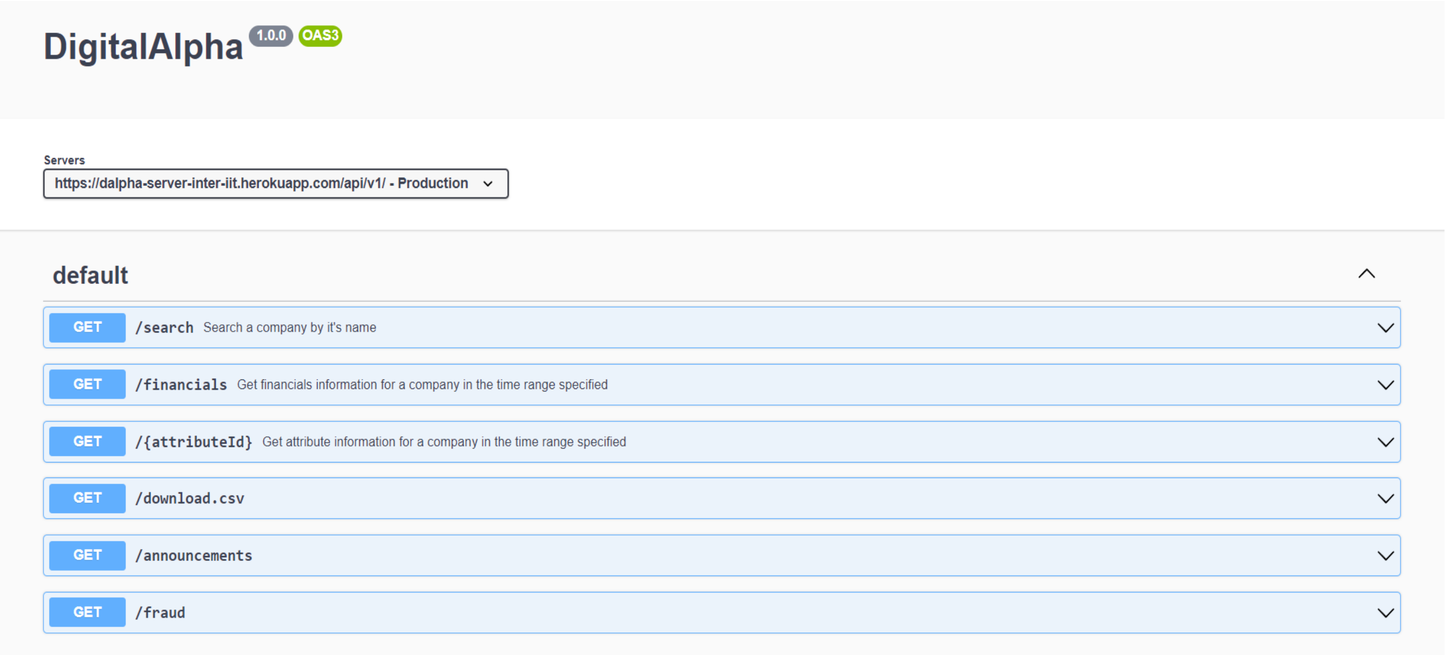
https://dalpha-server-inter-iit.herokuapp.com/api-docs

Architecture

The backend server is a separate service from the frontend. It is developed using an API first approach with OpenAPI 3.0 specification. The codebase is generated using OpenAPI Generator with the nodejs-express-server generator. Database used is MongoDB

API

The server exposes a REST API, which uses JSON for encoding of information. The API is documented at API docs:



DigitalAlpha 1.0.0 OAS3

Servers

https://dalpha-server-inter-iit.herokuapp.com/api/v1/ - Production

default

GET /search Search a company by it's name

GET /financials Get financials information for a company in the time range specified

GET /{attributeId} Get attribute information for a company in the time range specified

GET /download.csv

GET /announcements

GET /fraud

- **/search:**
  It searches the database for a particular company by name and returns the company's CIK in the EDGAR database and other general details of the company.

- **/financials:**
  It returns the financial information scraped from balance sheets and other places to the frontend in an easy-to-use format.

- **/{attributeId}:**
  It returns details of a particular SaaS growth metric for a company within a specified time range. The metric can be directly be scraped from the SEC filing website or can be calculated using some other scraped metrics.

  The available metrics:
  - Return on Equity
  - Net Inventory
  - Liabilities
  - Financial Leverage
  - Operating Income
  - Non-Operating Income Expense
  - Depreciation
  - Income Taxes Extraordinary Items Non-Controlling Interest
  - Net Income Loss

- **/download.csv:**
  It allows the users to download all the available scraped data into a CSV file which can be processed using any standard spreadsheets software like Microsoft Excel or Google Sheets.

- **/announcements:**
  Returns the announcements made by the company by filing 8K forms. The forms are scraped from the SEC website and undergo sentiment analysis to make it easier for investors to figure out the impact of the announcement on the company.

- **/fraud:**
  It returns the company's fraud percentage score.

Public API

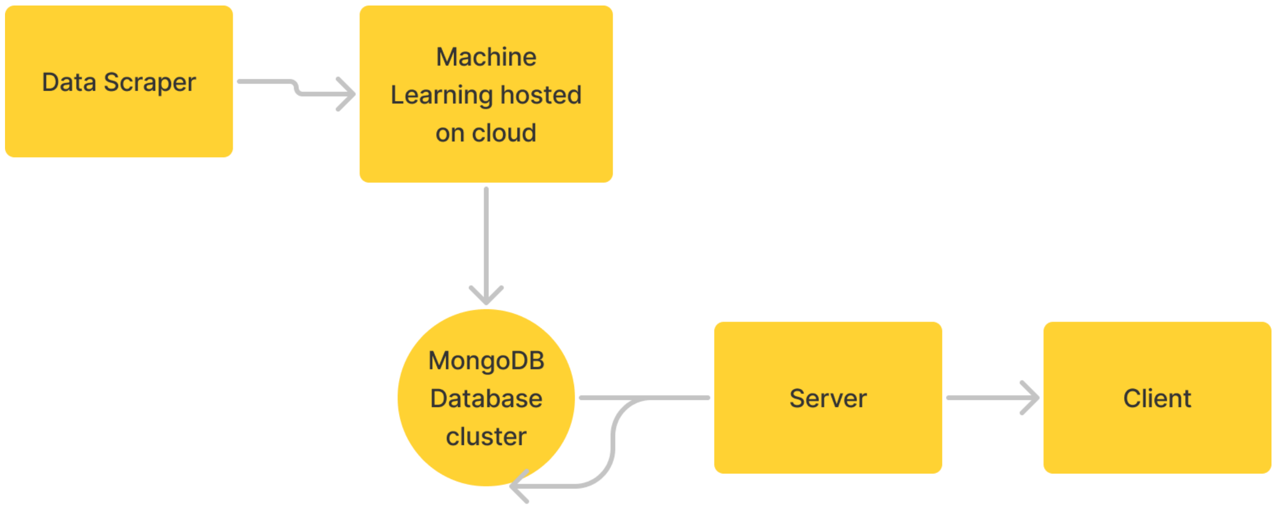
Since the EDGAR database doesn't have any API to access information, we have made our API publicly usable. The API comes with easy read Swagger generated documentation using our OpenAPI specification file.

Portability & Scalability

The server is a NodeJS server that can be easily dockerized using a Linux base image and can be ported to any cloud environment at ease. Since the application is stateless, it can be easily clustered using Kubernetes and deployed on any standard cloud hosting provider.

Hosting

Currently, the server is hosted on Heroku's Free Tier and the database on MongoDB Atlas. As mentioned earlier, the server can be hosted on any standard cloud hosting provider.



Frontend Development

https://dalpha.herokuapp.com/

Architecture

- The front-end is kept separate from the backend.
- The codebase is written using Next.js (a framework of React.js)
- It is hosted in Heroku. View Site ↗

UI/UX Specifications

- A uniform Material UI design is maintained across all the pages and related components.
- recharts (a composable charting library built on React components) are used for the purpose of data visualization in the form of line graphs or connected scatter plots.

Pages Included

The client-side rendering comprises of 6 pages with underlying components inside them:

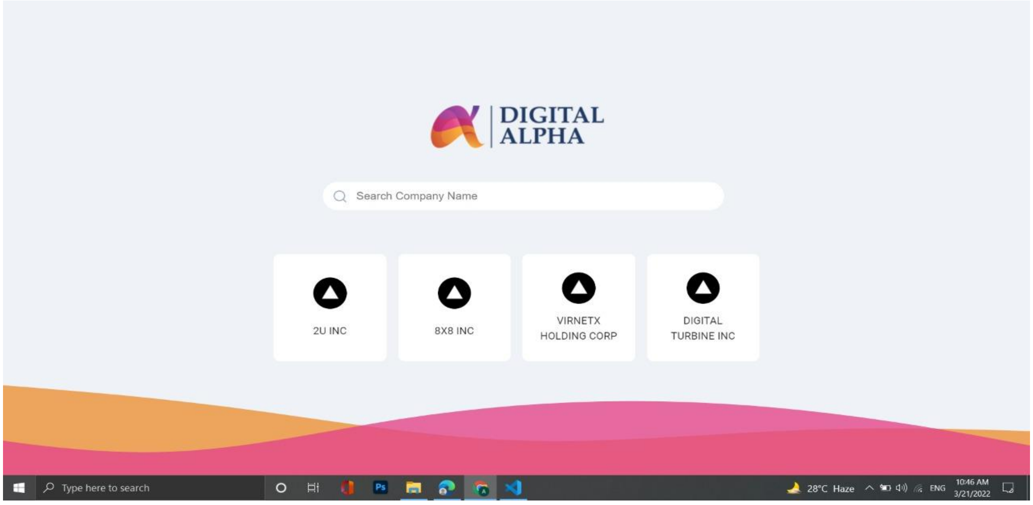
1. Home Page

2. Company Info Page
3. Company Metrics Page
4. Company Financials Page
5. Company Announcements Page

Homepage

It comprises:

- A search bar to search for the company whose details are required to be viewed.
- A list of recommended Companies is at the bottom of the search bar.



All the other pages comprise 3 main sections:

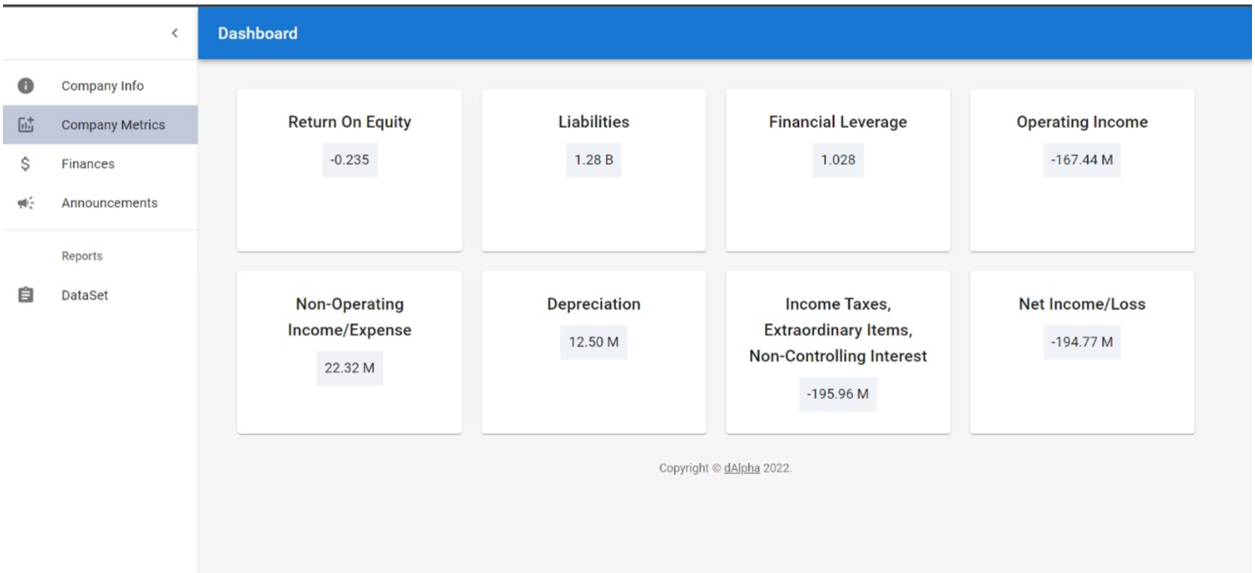
- Sidebar containing menu options for routing
- Title section displaying the name of the Company of which the page is a part
- A component section gives details for which the page is intended

Company Info Page:

Shows basic information about the Company including its id and name.

Company Metrics Page:

Shows details of the various SaaS metrics. It plots the attribute values as a function of time



Return On Equity: -0.235

Liabilities: 1.28 B

Financial Leverage: 1.028

Operating Income: -167.44 M

Non-Operating Income/Expense: 22.32 M

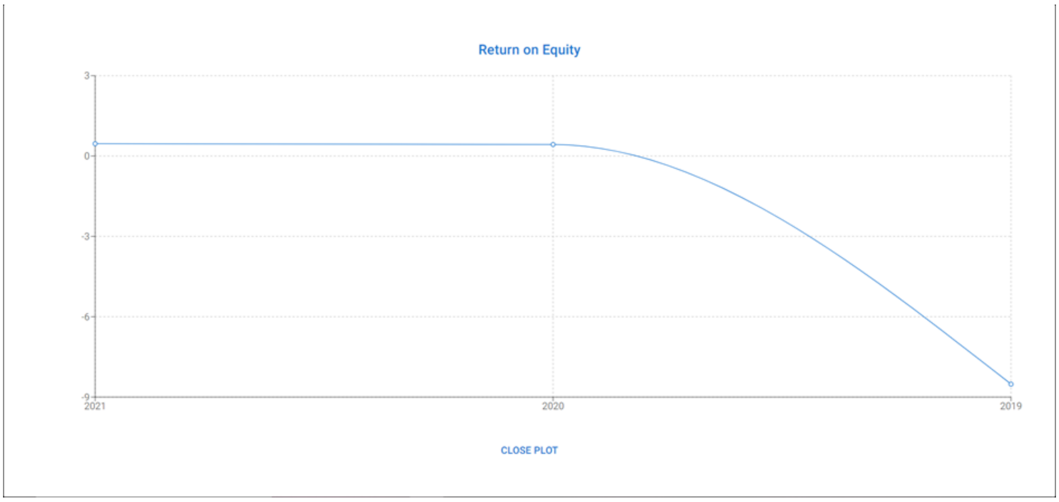
Depreciation: 12.50 M

Income Taxes, Extraordinary Items, Non-Controlling Interest: -195.96 M

Net Income/Loss: -194.77 M

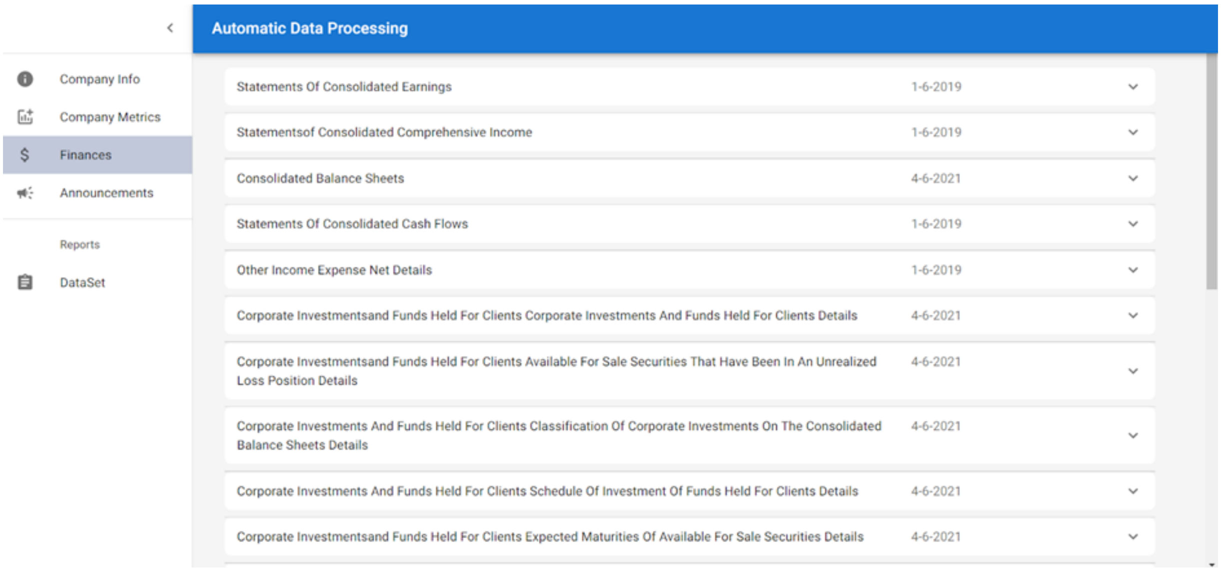
Copyright © dAlpha 2022.

On clicking a particular metric, the charts can be viewed.



Company Financial Page:

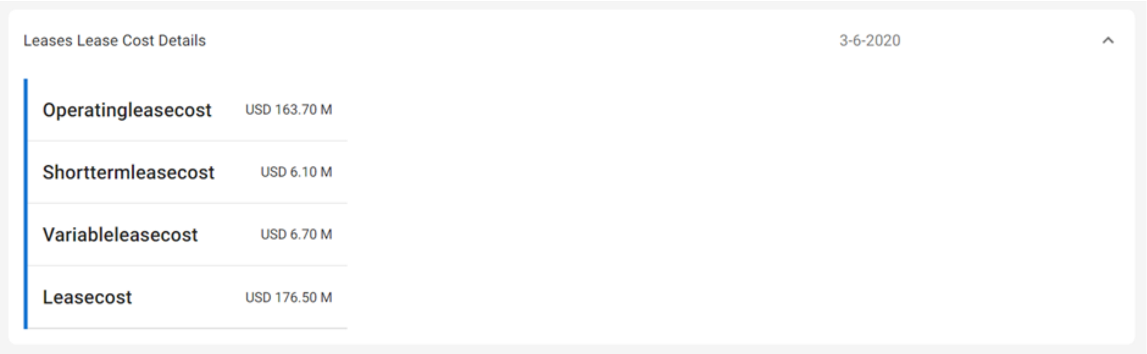
Shows the various financial attributes and sub-attributes of the company. Implemented in the form of an accordion.



Automatic Data Processing

Statements Of Consolidated Earnings	1-6-2019
Statementsof Consolidated Comprehensive Income	1-6-2019
Consolidated Balance Sheets	4-6-2021
Statements Of Consolidated Cash Flows	1-6-2019
Other Income Expense Net Details	1-6-2019
Corporate Investmentsand Funds Held For Clients Corporate Investments And Funds Held For Clients Details	4-6-2021
Corporate Investmentsand Funds Held For Clients Available For Sale Securities That Have Been In An Unrealized Loss Position Details	4-6-2021
Corporate Investments And Funds Held For Clients Classification Of Corporate Investments On The Consolidated Balance Sheets Details	4-6-2021
Corporate Investments And Funds Held For Clients Schedule Of Investment Of Funds Held For Clients Details	4-6-2021
Corporate Investmentsand Funds Held For Clients Expected Maturities Of Available For Sale Securities Details	4-6-2021

The sub-attributes can be viewed by opening the accordion.

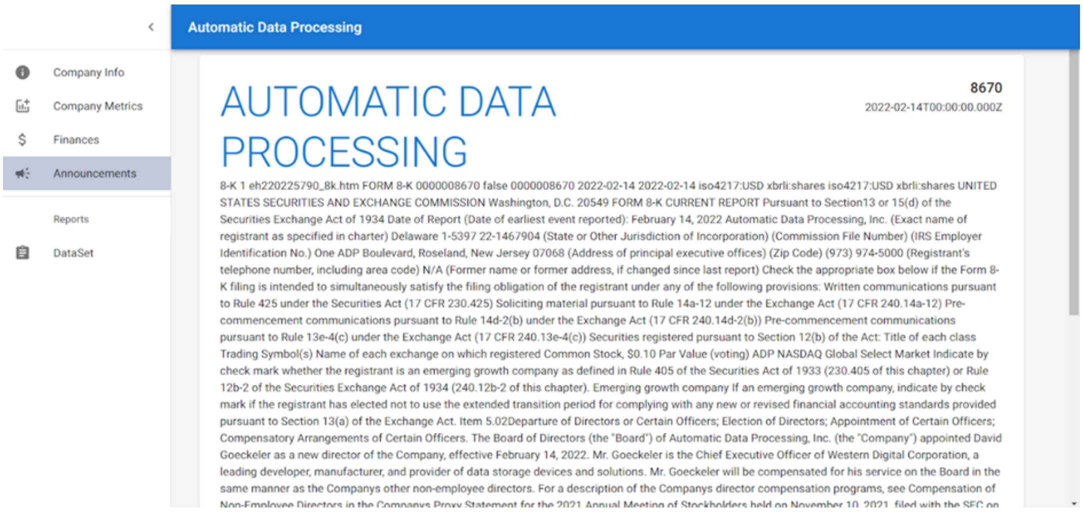


Leases Lease Cost Details 3-6-2020

Operatingleasecost	USD 163.70 M
Shorttermleasecost	USD 6.10 M
Variableleasecost	USD 6.70 M
Leasecost	USD 176.50 M

Company Announcement Page:

Shows the announcements made by the Company by filling 8K forms.

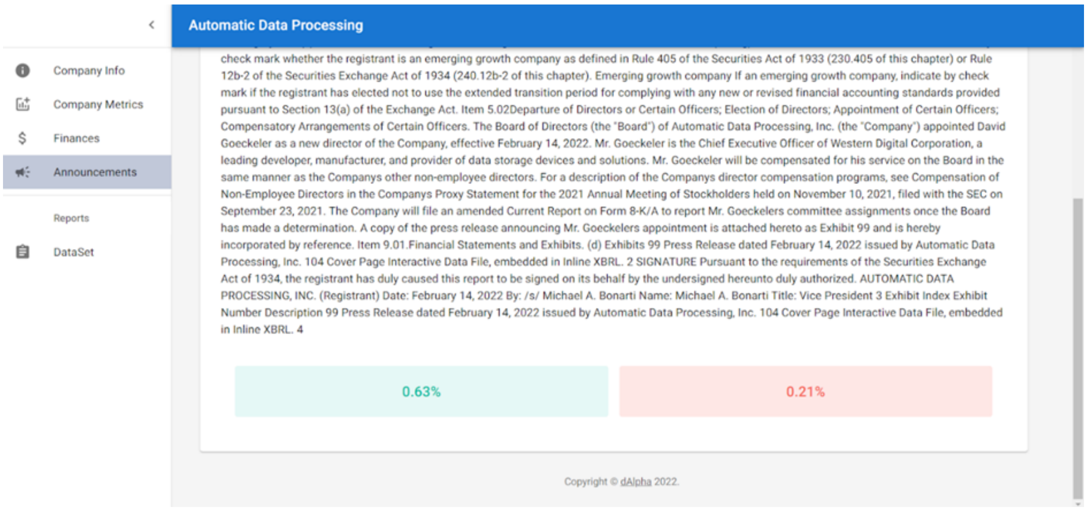


AUTOMATIC DATA PROCESSING

8670

2022-02-14T00:00:00.000Z

8-K 1 eh220225790_8k.htm FORM 8-K 0000008670 false 0000008670 2022-02-14 2022-02-14 iso4217:USD xbrli:shares iso4217:USD xbrli:shares UNITED STATES SECURITIES AND EXCHANGE COMMISSION Washington, D.C. 20549 FORM 8-K CURRENT REPORT Pursuant to Section13 or 15(d) of the Securities Exchange Act of 1934 Date of Report (Date of earliest event reported): February 14, 2022 Automatic Data Processing, Inc. (Exact name of registrant as specified in charter) Delaware 1-5397 22-1467904 (State or Other Jurisdiction of Incorporation) (Commission File Number) (IRS Employer Identification No.) One ADP Boulevard, Roseland, New Jersey 07068 (Address of principal executive offices) (Zip Code) (973) 974-5000 (Registrant's telephone number, including area code) N/A (Former name or former address, if changed since last report) Check the appropriate box below if the Form 8-K filing is intended to simultaneously satisfy the filing obligation of the registrant under any of the following provisions: Written communications pursuant to Rule 425 under the Securities Act (17 CFR 230.425) Soliciting material pursuant to Rule 14a-12 under the Exchange Act (17 CFR 240.14a-12) Pre-commencement communications pursuant to Rule 14d-2(b) under the Exchange Act (17 CFR 240.14d-2(b)) Pre-commencement communications pursuant to Rule 13e-4(c) under the Exchange Act (17 CFR 240.13e-4(c)) Securities registered pursuant to Section 12(b) of the Act: Title of each class Trading Symbol(s) Name of each exchange on which registered Common Stock, $0.10 Par Value (voting) ADP NASDAQ Global Select Market Indicate by check mark whether the registrant is an emerging growth company as defined in Rule 405 of the Securities Act of 1933 (230.405 of this chapter) or Rule 12b-2 of the Securities Exchange Act of 1934 (240.12b-2 of this chapter). Emerging growth company If an emerging growth company, indicate by check mark if the registrant has elected not to use the extended transition period for complying with any new or revised financial accounting standards provided pursuant to Section 13(a) of the Exchange Act. Item 5.02Departure of Directors or Certain Officers; Election of Directors; Appointment of Certain Officers; Compensatory Arrangements of Certain Officers. The Board of Directors (the "Board") of Automatic Data Processing, Inc. (the "Company") appointed David Goeckeler as a new director of the Company, effective February 14, 2022. Mr. Goeckeler is the Chief Executive Officer of Western Digital Corporation, a leading developer, manufacturer, and provider of data storage devices and solutions. Mr. Goeckeler will be compensated for his service on the Board in the same manner as the Companys other non-employee directors. For a description of the Companys director compensation programs, see Compensation of Non-Employee Directors in the Companys Proxy Statement for the 2021 Annual Meeting of Stockholders held on November 10, 2021, filed with the SEC on



check mark whether the registrant is an emerging growth company as defined in Rule 405 of the Securities Act of 1933 (230.405 of this chapter) or Rule 12b-2 of the Securities Exchange Act of 1934 (240.12b-2 of this chapter). Emerging growth company If an emerging growth company, indicate by check mark if the registrant has elected not to use the extended transition period for complying with any new or revised financial accounting standards provided pursuant to Section 13(a) of the Exchange Act. Item 5.02Departure of Directors or Certain Officers; Election of Directors; Appointment of Certain Officers; Compensatory Arrangements of Certain Officers. The Board of Directors (the "Board") of Automatic Data Processing, Inc. (the "Company") appointed David Goeckeler as a new director of the Company, effective February 14, 2022. Mr. Goeckeler is the Chief Executive Officer of Western Digital Corporation, a leading developer, manufacturer, and provider of data storage devices and solutions. Mr. Goeckeler will be compensated for his service on the Board in the same manner as the Companys other non-employee directors. For a description of the Companys director compensation programs, see Compensation of Non-Employee Directors in the Companys Proxy Statement for the 2021 Annual Meeting of Stockholders held on November 10, 2021, filed with the SEC on
September 23, 2021. The Company will file an amended Current Report on Form 8-K/A to report Mr. Goeckelers committee assignments once the Board has made a determination. A copy of the press release announcing Mr. Goeckelers appointment is attached hereto as Exhibit 99 and is hereby incorporated by reference. Item 9.01.Financial Statements and Exhibits. (d) Exhibits 99 Press Release dated February 14, 2022 issued by Automatic Data Processing, Inc. 104 Cover Page Interactive Data File, embedded in Inline XBRL. 2 SIGNATURE Pursuant to the requirements of the Securities Exchange Act of 1934, the registrant has duly caused this report to be signed on its behalf by the undersigned hereunto duly authorized. AUTOMATIC DATA PROCESSING, INC. (Registrant) Date: February 14, 2022 By: /s/ Michael A. Bonarti Name: Michael A. Bonarti Title: Vice President 3 Exhibit Index Exhibit Number Description 99 Press Release dated February 14, 2022 issued by Automatic Data Processing, Inc. 104 Cover Page Interactive Data File, embedded in Inline XBRL. 4

0.63%

0.21%

Copyright © dAlpha 2022.

*There is an additional **download button** at the sidebar of each page that downloads the entire data of the current company in the form of a CSV*

Interaction with Backend

On submitting the search form of the Home Page, the request is sent to the route: `/search`

- The following when loaded makes an initial request as follows:
  - Company metrics Page: `/{attributeId}`
  - Company financials Page: `/financials`
  - Company announcements Page: `/announcements`
- The download button makes a request to the API endpoint `/{download.csv}`

Scalability

The database used here(MongoDb) is scalable as its data is not coupled relationally. It can add multiple servers in parallel which increases the scalability.
We have used NextJs for the client-side, which uses SSR(Server Side Rendering) and ISR(Incremental Static Regeneration). This renders the required pages on the client side in the form of static pages using its GetStaticProps and GetStaticPaths methods and has iSR which makes the required changes on the client side if there had been a change on the server side given a revalidation time.