**Abstract**

As undergrad or grad students, we are in throes of finding a summer internship or a post-college job search. The process can be exhausting and time taking considering the research that goes behind finding the perfect role and more importantly the right company. Hence our application is aimed at making the ‘research’ simpler for people looking for jobs.

The application will enable the user to view a range of stats of a company and filter out companies that match their interests. Eventually, the user can use it to find jobs at these companies and view recommendations of similar job opportunities.

**Introduction and project goals**

This application will enable the user to view a range of stats of a company and filter out companies that match their interests. The functionalities in the web app like viewing the latest news and sentiments will help user gain a holistic view about the reputation of the company. Eventually, the user can use it to find jobs at these companies and view recommendations of similar job opportunities.

**Data sources and technologies used**

We choose to work with FMP API as it is freely and easily available, while also it is guaranteed to be consistent and accurate. In addition to the raw financial data teh FMP API also provides access to a host of other interesting data about companies like what are a company's stock peers (essentially mapping how companies are related to each other) and what are the social sentiment of a company (with data from a host of social media sites like twitter, reddit, etc.).

We will use the Requests package in python to query for the information corresponding to all the companies available through the FMP API and collect the attributes specified above to fill the table. This table then can be joined with the table from the Indeed.com scrapping to perform various queries as desired.

Indeed.com is one of the top websites job seekers prefer for finding work opportunities. Due to the availability of up-to-date information displayed in an organized and structured fashion, Indeed will serve as a befitting source of job/internships data for our project.  
Indeed came out as the top choice amongst Linkedin and Glassdoor due to a well defined html structure and free availability of information without a paywall or requirements for login.

Technologies used will include Python’s requests library, Selenium, Beautiful Soup, MySQL, node.js, and react.js. Requests and Beautiful Soup will be used to build the database, MySQL will be used to handle queries, Node will be used to launch the backend, React will be used to create the UI, and Selenium will be used for testing.

**Relational Schema**

**Companies**(id, symbol, name, price, exchange, exchangeShort, type)

**CompanyInformation**(id,symbol volAvg, marketCap, currency, phone, industry, website, description, ceo, sector, country, address, image, zip, state, city, fullTimeEmployees)

FOREIGN KEY (id) REFERENCES Companies(id)

**CompanyNews**(id,url, publishedDate, title, text, imageLink, website)

FOREIGN KEY (id) REFERENCES Companies(id)

**Sentiment**(companyID, date, absoluteIndex, relativeIndex, generalPerception, sentiment, redditPostMentions, redditPostSentiment, redditCommentMentions, redditCommentSentiment, tweetMentions, tweetSentiment, stocktwitsPostMentions, stocktwitsPostSentiment, yahooFinanceCommentMentions, yahooFinanceCommentSentiment)

FOREIGN KEY (companyID) REFERENCES Companies(id)

**Peers(**companyID, peerID)

FOREIGN KEY (companyID) REFERENCES Companies(id)

**IndeedJobs(**indeedJobLink, jobTitle, indeedCompanyName, rating, jobLocation, jobType, shortDescription, payPerHour, companyID, companyName)

FOREIGN KEY (companyID) REFERENCES Companies(id)

FOREIGN KEY (companyName) REFERENCES Companies(name)

**Description of system architecture**

Below are descriptions of our Career portal service web pages with information about what the respective pages accomplish:

* **Home Page:** This is the first page the user will end up on when they visit our web application. The home page has a search functionality which lets the user search for companies by applying filters like: Name of Company, Number of employees, market capitalization, sector etc. The search will list all the companies that match the description.
* **The Company Page:** Once the user selects a company from the search in the homepage they will be directed to the company page which will be dynamically generated. This page will hold general information about the company like description of the work it does, CEO, contact details like phone number, number of employees, website link etc.
* **The News Page:** From the company page the user can navigate to the news page. The news page will fetch the latest news from the FMP database and update the page. Another added functionality is we will also fetch news of similar companies and display for the user to get a sense how the company is performing compared to peers.
* **The Sentiment Page:** We will display the social media information from FMP. This page will have an interactive visualization where we fetch the sentiments of peers and give a visual representation of how the company is perceived in general.
* **The Job Search Page:** This will help user apply filters according to their preference and find relevant jobs. The application will also recommend similar jobs.

**How we addressed required features**

We combined several datasets in a compelling and useful portal, one of which need to be scraped and   
cleaned.   
   
We generated complicated queries combining information from multiple tables and optimized the   
database performance by indexing, caching and carefully choosing join orders and selections.   
   
We used complex architecture incorporating many current, popular and relevant technologies:   
.

React, Nodejs, Selenium, Beautiful Soup, MySQL

**Performance evaluation**

Early in our project planning process, we observed that some of the attributes did not have atomic values and were not in I NF. in our ER diagram that some data was in 2NF. We manipulated our schema to ensure all our relations are in 3NF, and subsequently there is no redundancy in the tables after redistributing our data.

**Technical Challenges and How They were Overcome**

The first challenge we faced was using complicated data sources. While Kaggle was an easy upsert into   
our production database (MySQL), the rest of our data   
came from unstructured web sources and required us   
to recursively scrape the various player profiles that   
we would need for our player page, roster on our team   
page, and most importantly our battle page.   
   
Next, we faced substantial difficulty joining the   
different datasets. For example, some of the players   
names formats varied from our different data sources   
and therefore needed to be cleaned during the   
scraping process. This required us to make decisions   
about how best to parse player names so that we could   
link, for example, one particular player’s statistics   
with their image.

Appendix D

Appendix E

SQL Queries

1. Returns an array of companies according to the search criteria specified by the user.

WITH tmp1 AS

(SELECT symbol, companyName, fullTimeEmployees, mktCap

FROM CompanyInformation

WHERE companyName LIKE '%${cmpName}%' and

fullTimeEmployees BETWEEN ${numEmployeesLow} AND ${numEmployeesHigh}

AND mktCap BETWEEN ${mktcapLow} AND ${mktcapHigh}),

tmp2 AS (SELECT s.symbol, tmp1.companyName, tmp1.fullTimeEmployees, tmp1.mktCap, s.sentiment

FROM CompanySentiments s

JOIN tmp1

ON tmp1.symbol= s.symbol

WHERE sentiment BETWEEN ${sentiLow} AND ${sentiHigh})

SELECT \*, COUNT(jobLink) as JobCount

FROM IndeedJobs i

JOIN tmp2

ON tmp2.symbol=i.companySymbol

GROUP BY i.companySymbol

HAVING COUNT(jobLink)> ${jobNum}

LIMIT ${pagesize} OFFSET ${offset};

1. Returns jobs according to the search criteria specified by the use

WITH TT as (

SELECT \*

FROM CompanyInformation CI

JOIN IndeedJobs I ON I.companySymbol = CI.symbol

WHERE CI.industry LIKE '%${industry}%' AND CI.sector LIKE '%${sector}%' AND

CI.companyName LIKE '%${cmpName}%' AND I.jobType LIKE '%${jobType}%' AND

(I.companyRating BETWEEN ${ratingLow} AND ${ratingHigh})

AND (CI.fullTimeEmployees BETWEEN ${numEmployeesLow} AND ${numEmployeesHigh})

AND I.jobTitle LIKE '%${jobTitle}%'

ORDER BY CI.companyName

)

SELECT \* FROM TT LIMIT 50;

1. Returns company information according to the symbol specified by user.

SELECT \*

FROM CompanyInformation

WHERE symbol LIKE '${company}

1. Returns company information of the peers of the specified company

with t as (select peerID from Peers where symbol like '%${company}%')

select companyName, peerID

from CompanyInformation ci JOIN t on ci.symbol=t.peerID LIMIT ${pagesize} OFFSET ${offset};

1. Returns news article relating to the company with the matching symbol as well as relating to its peers sorted by the published date in descending order.

WITH Temp AS ( WITH T1 AS (

SELECT DISTINCT peerID, 'PEER' as cmpRel

FROM Peers

WHERE symbol LIKE '${company}'

UNION

SELECT DISTINCT peerID, 'SELF' as cmpRel

FROM Peers

WHERE peerID LIKE '${company}')

SELECT \*

FROM CompanyNews CN JOIN T1 ON CN.symbol = T1.peerID

ORDER BY publishedDate DESC)

SELECT symbol,

publishedDate,

title,

image,

site,

text,

url,

cmpRel

FROM Temp

ORDER BY publishedDate DESC

LIMIT ${offset}, ${pagesize};

1. Returns an aggregated sentiment rating for the company with the matching symbol as well as its peers and an average of all of its peers sentiment values

WITH T1 AS (SELECT DISTINCT peerID as ID

FROM Peers

WHERE symbol = '${company}' or peerID = '${company}'),

T2 AS (SELECT C.companyName, C.symbol, T1.ID, 2 as ord

FROM CompanyInformation C JOIN T1 ON C.symbol = T1.ID

WHERE symbol != '${company}'),

T3 AS (SELECT C.companyName, C.symbol, T1.ID, 0 as ord

FROM CompanyInformation C JOIN T1 ON C.symbol = T1.ID

WHERE symbol = '${company}'),

T23 AS (SELECT \*

FROM T2 UNION ALL

SELECT \*

FROM T3),

T4 AS (SELECT S.sentiment, S.relativeIndex, T1.ID

FROM CompanySentiments S JOIN T1 ON S.symbol = T1.ID),

T5 AS (SELECT 'Average of peers' AS companyName, 'AVG' AS symbol, AVG(S.sentiment) as sentiment, AVG(S.absoluteIndex) as relativeIndex, 1 as ord

FROM CompanySentiments S JOIN T1 ON S.symbol = T1.ID

WHERE T1.ID != '${company}')

SELECT \* FROM T5

UNION ALL

SELECT T23.companyName, T23.symbol, T4.sentiment, T4.relativeIndex, T23.ord

FROM T23 JOIN T4 ON T23.ID = T4.ID

ORDER BY ord;

1. Returns an array of job listings from the company with the matching symbol as well as from its peers.

WITH Temp AS (

WITH T1 AS (SELECT DISTINCT peerID , 'SELF' as cmpType

FROM Peers

WHERE peerID LIKE '${company}'

UNION ALL

SELECT DISTINCT peerID , 'PEER' as cmpType

FROM Peers

WHERE symbol LIKE '${company}'

)

SELECT \*

FROM IndeedJobs IJ JOIN T1 ON IJ.companySymbol = T1.peerID

ORDER BY postingDate DESC)

SELECT companySymbol,

searchCompany,

jobType,

jobCountry,

searchLink,

jobTitle,

jobLink,

jobCompany,

companyLink,

companyRating,

jobLocation,

shortDescription,

postingDate,

salary,

cmpType

FROM Temp

ORDER BY companySymbol

LIMIT ${offset},${pagesize}