



**ITC 6460 – Cloud Analytics**

**Final Project Report**

**Reddit Stock Analysis using AWS Cloud Analytics Services**

Submitted to:

**Prof. Sergiy Shevchenko**

**Submitted by**

**Group Husky 6**

Anubhav Saha

Ashishkumar Bidap

Ashlesha Kshirsagar

Priyanka Nagesh Adiga

Nikhil Sanjay Thorat

02/23/2021

**Abstract:**

Lately, a group of online traders collaborated on Reddit & took down the hedge funds by bidding up the stock price of GameStop. This unprecedented move by the online traders took the market by storm with continuous ups and downs in the stock value. Therefore, we decided to come up with a project that can analyze the discussions on Reddit for “/r/WallStreetBets” and give the users an idea about what are some of the stocks that are being discussed, sentiment of their discussion and further allowing the user to view the additional details about the required stock over an interactive dashboard.

***Keywords:*** *Amazon comprehend, amazon lex, visualization*

## **Content**

1.Introduction.....	4
2. Executive Summary .....	5
3. Requirements and Assumptions .....	6
4. AWS Services used.....	7
5.Architecture Diagram.....	8
6.Execution .....	8
6.1. Amazon Lex.....	8
6.2.Data Collection.....	12
6.3.Amazon Comprehend.....	15
6.4.Athena and Glue.....	20
6.5.Quicksight.....	22
7.Web interface.....	24
8. Challenges.....	25
9. Future scope .....	26
10.Conclusion .....	27
11.Reference.....	27

## **1.Introduction**

Big hedge funds short a stock and then trash talk the company to drive down the share price and make a ton of money. This has been the norm in Wall Street before a group of online traders decided to teach them a lesson by collaborating on Reddit and buying and holding GameStop shares, which is a struggling video game retailer against whom the hedge funds were betting. This unprecedented move by the online traders took the market by storm which caused the big hedge funds to lose billions of dollars. The situation got further escalated after the *Wallstreetbets* Reddit group got the backing from Elon Musk on Twitter, who is vocally against the practice of shorting of stocks. This caused GameStop's market value to increase to over \$24 billion from \$2 billion in a matter of days.[1]

The platform where most of these online traders trade, called "Robinhood, " decided to limit buying some of these stocks, which prompted further criticism and attracted much attention to this issue. Therefore, we decided to come up with a project that can analyze these discussions on Reddit and give us an idea about what are some of the stocks that are being discussed in these forums and what can be expected from these online traders in the upcoming days based on the Sentiment of their discussion. This can give stock traders valuable insights before they decide to buy or sell a stock.

## **2.Executive Summary**

To get the idea about the names of the stocks that are being discussed on the Reddit platform along with the associated Sentiment and other stock market trends, we built a chatbot using Amazon Lex that accepts user inputs and gives the user an option to analyze the Reddit discussions or view a dashboard with the latest stock market trends. The Lex bot uses a lambda function to make an API call that fetches the Reddit forum's required discussions and stores it in an S3 bucket. This data is then used by Amazon Comprehend using another Lambda function for analysis and detection of entities, that includes the names of organizations/stocks that are mentioned in the discussions and also the overall Sentiment of these discussions. These results are then stored back to the S3 bucket, from where the Lex bot uses another lambda function to display the results back to the user in the chat interface.

Alternatively, the user can choose to view the stock market trends and their detailed visualizations in a dashboard by providing that input to the Lex chat interface. The user is then provided with the URL to the dashboard created using QuickSight, which has multiple tabs with data visualizations of the market trends such as the average net change in the various sectors, market cap value by industry, etc. It also shows a word cloud of the key phrases that the people in the industry are talking about, all of which provide the necessary insights to an investor to make an informed decision while trading in the market.

### **3.Requirements and Assumptions**

#### **3.1. Requirements**

1. The solution delivered should be serverless, i.e., without having to set up and manage any servers or data warehouses.
2. Users of the application get an idea of what is being discussed on Reddit in context with the stock market.
3. Users accessing the application have seamless access to the application functionalities.

#### **3.2. Assumptions**

1. Redditors on the channel "/wallstreetbets" discuss only stocks and options trading.
2. Nasdaq has been the place for tech stocks to go public, and also a large group of fast-growing stocks is listed on Nasdaq; thus, we have considered Nasdaq stocks data in our analysis.
3. We are analyzing the top 100 trending posts of Reddit only assuming the top 100 posts are the ones which are highly upvoted and active on the "/WallStreetBets" channel
4. Publicly shareable quicksight dashboard links come with only premium subscription of Quicksight so currently only the users authorized with the dashboard would be able to view the dashboard

#### **4.AWS Services Used:**

1. **Amazon Comprehend:** Amazon Comprehend provides a pre-trained, machine learning NLP model, which will be used to extract Entity, a key phrase, Sentiment, and syntax.
2. **Lambda function:** It will execute the code in response to triggers such as changes in data, shifts in systems state, or user actions. Because Amazon S3 can directly trigger lambda function, we will build a real-time serverless data processing system.
3. **Amazon S3:** Filtered text data and extracted data will be stored in the S3 data lake.
4. **Amazon Lex:** Provides an interface to create conversational chatbots.
5. **Quick Sight:** Quick sight is a serverless, embeddable Business Intelligence tool that can create interactive dashboards using the insights acquired from machine learning algorithms. Quick sight dashboard can be easily shared on various websites, devices, and portals.
6. **Amazon Athena:** Athena is a serverless service that does not need any infrastructure to create, manage, or scale data sets. It works directly on top of Amazon S3 data sets. And creates external tables.
7. **Amazon Glue:** AWS Glue provides both visual and code-based interfaces to make data integration easier. It Populates the AWS Glue Data Catalog with table definitions from scheduled crawler programs also detects schema changes and adapts based on preferences

## 5. Architecture Diagram

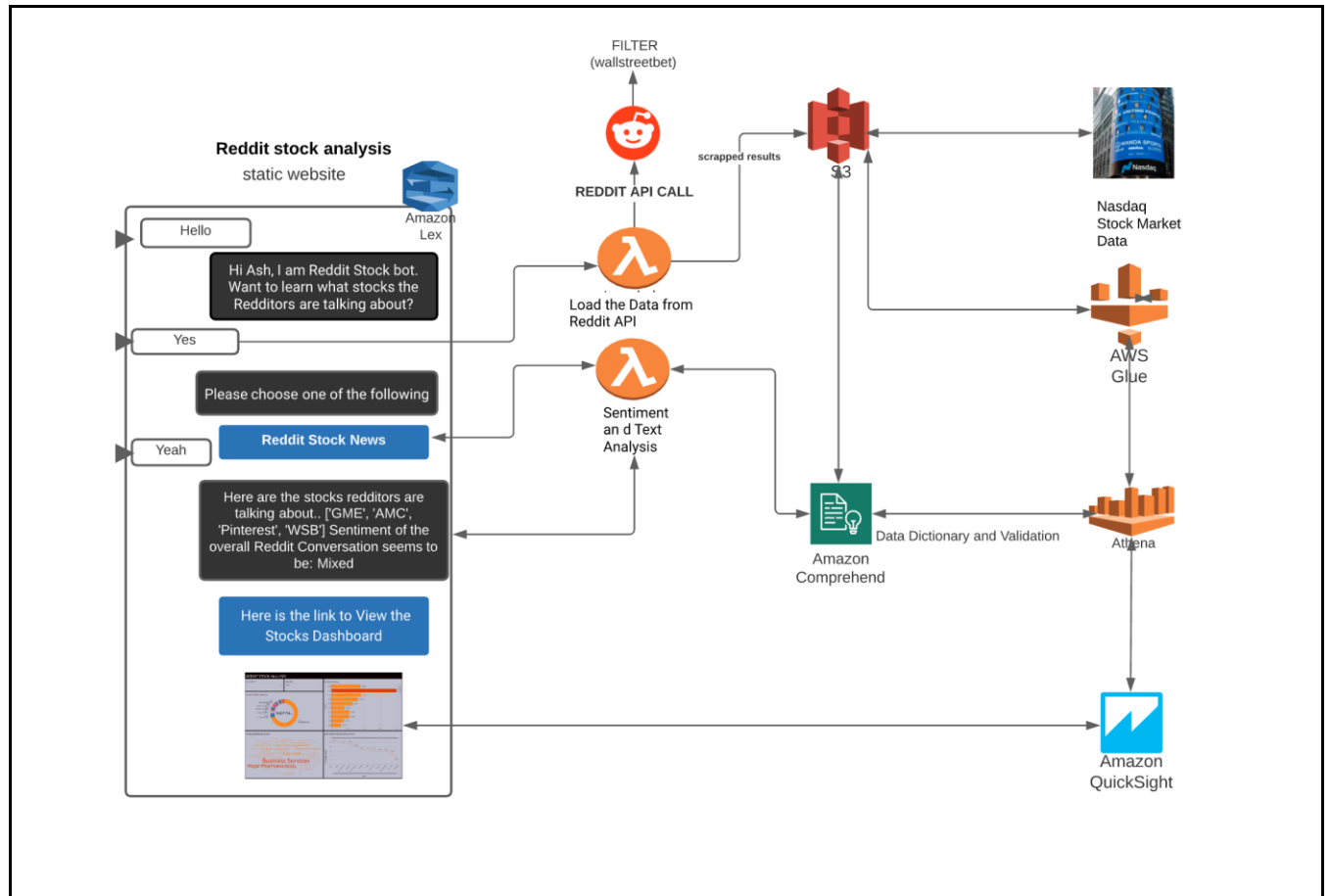


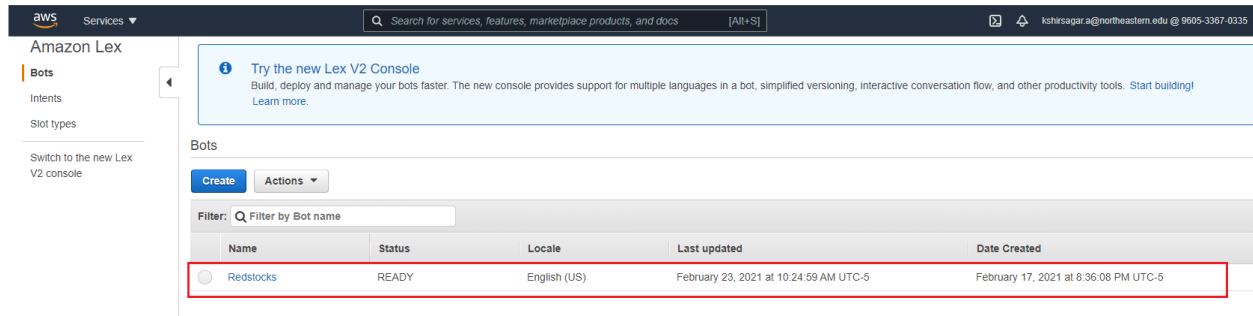
Figure 1. Architecture Diagram of Reddit chatbot

## 6. Execution

### 6.1. Amazon Lex

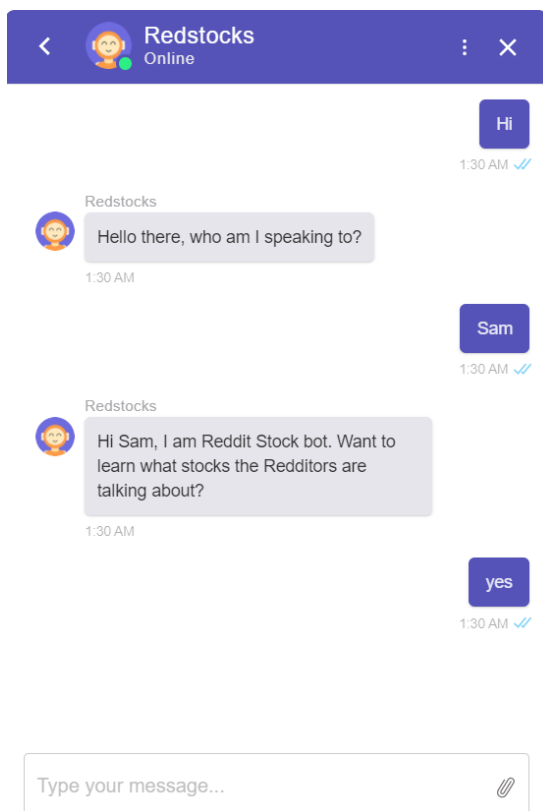
The proposed project utilizes a conversational interface, typically a chatbot, to enable the end-users to access the latest stock market information. Amazon lex service is used for implementing this. This service is used to acquire the benefits of natural language understanding it offers so as to recognize the intent of the text and enable highly user engaging and life-like conversations with end-users.





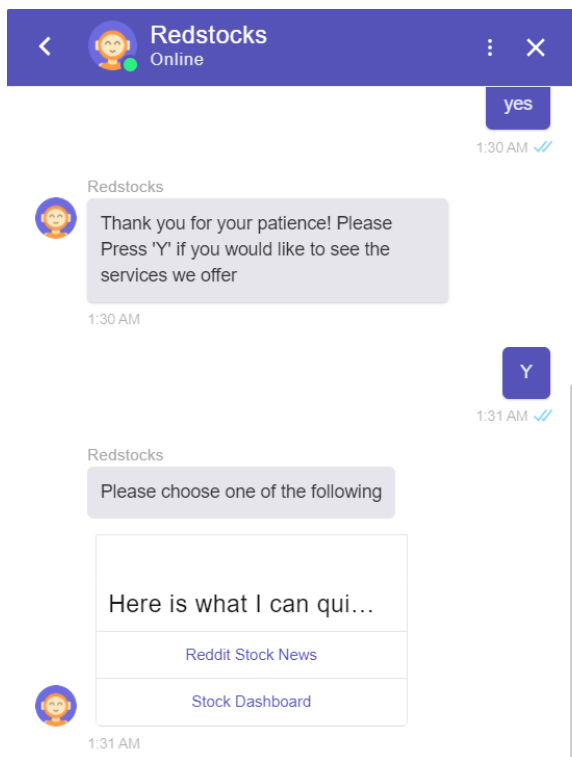
In this project, we have formulated seven intents in order to formulate an effective functioning informative chatbot. The following section provides a brief on each intent.

**1 HelloIntent:** As the name suggests, this intent handles the greeting aspect of the chatbot and provides a brief introduction about what the bot is about and uses one or more customary greeting responses to fulfill the intent.



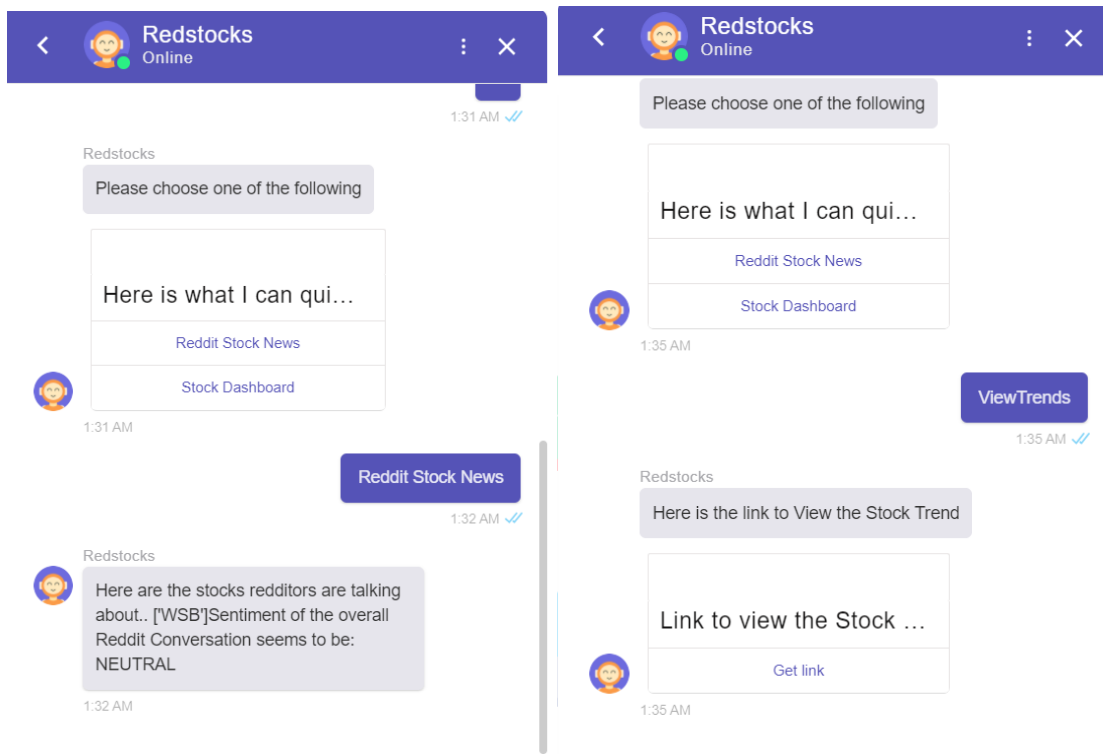
**2 AnalyzeIntent:** This is invoked when the end-user seeks stock market information. Intent seeks for user confirmation on his intent, and this intent is fulfilled via invoking a lambda function that typically collects the required data from Reddit to serve the user with the relevant information.

**3 ApprovalIntent:** This intent is invoked when the user confirms his intentions to the information he needs. To fulfill the intent, the user is presented with two options, 'Reddit Sock News' and 'Stock Dashboard'.



**4 ComprehendIntent:** This intent is fired when the user opts for viewing the latest reddit news related to the stock market. The intent is fulfilled upon calling a lambda function that processes the data acquired from reddit and provides a response. The response of this intent contains most used names of the stocks in typical reddit conversations along with the Sentiment associated with these conversations.

**5 DashboardIntent:** This intent is fired when the user opts for viewing the latest stock market dashboard. The intent is fulfilled by providing an url that leads the end-user to the dashboard formulated based on the data from Nasdaq.



**1.6 ThankYouIntent:** As the name suggests, this intent is fired when the user shows a general appreciation. The intent is fulfilled via providing the user with a brief of other services the chatbot offers.

**1.7 DenyIntent:** This intent is fired when a user rejects or expresses a disapproval for any of the above-mentioned intents. The intent is fulfilled with a general response.

## 6.2. Data Collection

### 6.2.1. Collecting Text Data from Reddit using AWS Lambda.

This section collects the data for the hot posts from an external system "Reddit" by accessing the reddit's open api for the reddit channel "r/wallstreetbets". The collected data is further transformed into a flat file which includes the top 200 hot posts on the channel with the following attributes created\_utc, downs, id, kind, link\_flair\_css\_class, score, selftext, title, ups, upvote\_ratio. Upon successful execution of the lambda function the flat file is loaded to the S3 bucket for further processing . Below is the Lambda function code we used to accomplish the data collection from reddit.

```
import json
import requests
import pandas as pd
from datetime import datetime
from io import StringIO
import boto3

# authenticate API
client_auth = requests.auth.HTTPBasicAuth('XXXX', 'XXXX')
data = {'grant_type': 'password', 'username': 'XXXX', 'password': 'XXXX'}
headers = {'User-Agent': 'mybot/0.0.1'}
# send authentication request for OAuth token
res=
requests.post('https://www.reddit.com/api/v1/access_token', auth=client_auth, data=data,
headers=headers)
TOKEN = f"bearer {res.json()['access_token']}"
headers = {**headers, **{'Authorization': TOKEN}}

def df_from_response(res):
    # initialize temp dataframe for batch of data in response
    df = pd.DataFrame()
    # loop through each post pulled from res and append to df
    for post in res.json()['data']['children']:
        df = df.append({
            'subreddit': post['data']['subreddit'],
            'title': post['data']['title'],
            'selftext': post['data']['selftext'],
            'upvote_ratio': post['data']['upvote_ratio'],
```

```

        'ups': post['data']['ups'],
        'downs': post['data']['downs'],
        'score': post['data']['score'],
        'link_flair_css_class': post['data']['link_flair_css_class'],
        'created_utc':
datetime.fromtimestamp(post['data']['created_utc']).strftime('%Y-%m-%dT%H:%M:%SZ'),
        'id': post['data']['id'],
        'kind': post['kind']
    }, ignore_index=True)
    return df

def lambda_handler(event, context):
    data = pd.DataFrame()
    params = {'limit': 100}
    print("Loading data to csv file..")
    try:
        for i in range(0,1):
            # make request
            res = requests.get("https://oauth.reddit.com/r/wallstreetbets/hot",
                               headers=headers,params=params)

            #get dataframe from response
            new_df = df_from_response(res)
            #take the final row (oldest entry)
            row = new_df.iloc[len(new_df)-1]
            #create fullname
            fullname = row['kind'] + '_' + row['id']
            #add/update fullname in params
            params['after'] = fullname
            #append new_df to data
            data = data.append(new_df, ignore_index=True)

    except:
        pass
    print("Loading data to S3..")
    #load the dataframe as .csv file to S3
    csv_buffer = StringIO()
    data.to_csv(csv_buffer)
    s3_resource = boto3.resource('s3')
    s3_resource.Object('output-reddit-data',
                        'output.csv').put(Body=csv_buffer.getvalue())
    print("Successful")
    result = {
        "sessionAttributes": {},
        "dialogAction": {
            "type": "Close",
            "fulfillmentState": "Fulfilled",
            "message": {
                "contentType": "PlainText",
                "content": "Thank you for your patience! Please Press 'Y'
if you would like to see the services we offer"

```

```
        # "content": "Successfully dumped the new posts from the  
'wallstreetbets' reddit channel.Excited to see the results ?"  
    }  
}  
}  
}  
return result
```

In order to make sure the AWS Lambda function communicates with the S3 bucket for loading the data, we created an IAM role as shown below which provides the read write access to the lambda function with the specified S3 bucket and further allows the cloudwatch to monitor the lambda function execution logs.

### IAM Roles and Policy associated with Lambda Function

The screenshot displays the AWS IAM console interface. At the top, there is a search bar and a user profile. The main content area shows the details of the 'Lambda\_S3GetPutAccess' role. The 'Summary' tab is active, displaying the role's ARN, description, and creation time. The 'Permissions' tab is also visible, showing the attached policies: 'LambdaS3PutGetPolicy' and 'AWSLambdaBasicExecutionRole'.

Policy name	Policy type
LambdaS3PutGetPolicy	Managed policy
AWSLambdaBasicExecutionRole	AWS managed policy

### 6.2.2 Collecting Raw Stock Data from Nasdaq

In this section we are loading a dataset of stock listings manually to the S3 bucket which includes the data attributes like Symbol, Name, Last Sale, Net Change, % Change, Market Cap, Country, IPO Year, Volume, Sector which would be further used as a data dictionary for refining our results and also as part of our analysis for the dashboard visualizations.

Below is the S3 bucket where the data files are loaded.

Amazon S3 > output-reddit-data

**output-reddit-data**

Objects | Properties | Permissions | Metrics | Management | Access Points

**Objects (2)**

Objects are the fundamental entities stored in Amazon S3. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

[Refresh](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)

Find objects by prefix

<input type="checkbox"/>	Name	Type	Last modified	Size
<input type="checkbox"/>	<a href="#">nasdaq_screener.csv</a>	csv	February 23, 2021, 12:22:47 (UTC-05:00)	507.8 KB
<input type="checkbox"/>	<a href="#">output.csv</a>	csv	February 23, 2021, 10:19:21 (UTC-05:00)	59.7 KB

Symbol	Name	Last Sale	Net Change	% Change	Market Cap	Country	IPO Year	Volume	Sector	Industry
A	Agilent Technologies Inc. Comm	\$127.68	-2.44	-1.88%	38992112463	United States	1999	1941522	Capital Goods	Biotechnology: Laboratory Analytical
AA	Alcoa Corporation Common Stoc	\$21.12	-0.81	-3.69%	3927856817		2016	3545632	Basic Industries	Aluminum
AACG	ATA Creativity Global American C	\$5.75	-0.27	-4.49%	180265381	China		1029362	Consumer Services	Other Consumer Services
AACQ	Artius Acquisition Inc. Class A Coi	\$12.78	-0.72	-5.33%	1157388750	United States	2020	5716686	Finance	Business Services
AACQU	Artius Acquisition Inc. Unit	\$13.91	-1.09	-7.27%	0	United States	2020	733879	Finance	Business Services
AACQW	Artius Acquisition Inc Warrant	\$3.56	-0.39	-9.87%	0	United States	2020	1818381	Finance	Business Services
AAIC	Arlington Asset Investment Corp	\$3.94	-0.05	-1.25%	131705087	United States		201427	Consumer Services	Real Estate Investment Trusts
ABB	ABB Ltd Common Stock	\$29.68	0.31	1.06%	60275158136	Switzerland		1517465	Consumer Durables	Electrical Products

### 6.3. Text Analytics using Amazon Comprehend

In this section we take the reddit data from S3 which was loaded from API and send it to amazon comprehend. Amazon Comprehend uses a pre-trained model to examine and analyze a document or set of documents to gather insights about it. and doesn't require training data. In this section we have used 'detect entities' and 'detect sentiment' to get the name of organizations from the text and overall Sentiment of the text respectively. Each Entity also has a score that indicates the level of confidence that Amazon Comprehend has that it correctly detected the entity type. In this project

we have used only 'ORGANIZATION' type which will detect the organizations/companies in the reddit text with entity score more than 95% confidence level.

```
import boto3
from pprint import pprint
from csv import reader
import pandas as pd
import os,sys,uuid
import time
from urllib.parse import unquote_plus

def split_list(datatitle):
    half = len(datatitle)//2
    list_1 = datatitle[:half]
    list_2 = datatitle[half:]
    return list_1,list_2

def lambda_handler(event, context):

    try:
        s3_client = boto3.client('s3')
        bucket_name = "output-reddit-data"
        s3_file_name = "output.csv"

        resp = s3_client.get_object(Bucket=bucket_name, Key=s3_file_name)
        df_s3_data = pd.read_csv(resp['Body'], sep=',')
        title = list(df_s3_data['title'])

        analysis_list_1,analysis_list_2 = split_list(title)
        analysis_string_1 = ' '.join(analysis_list_1)
        analysis_string_2 = ' '.join(analysis_list_2)

comprehend = boto3.client("comprehend")

        #Extracting entities using comprehend
        entities_1 = comprehend.detect_entities(Text = analysis_string_1, LanguageCode = "en")
        entities_2 = comprehend.detect_entities(Text = analysis_string_2, LanguageCode = "en")

        #fetching only the entities with organization type
        entity_text = []
        for entities in entities_1['Entities']:
            if entities['Type'] == 'ORGANIZATION' and entities['Score'] >= 0.95:
                entity_text.append(entities['Text'])

        for entities in entities_2['Entities']:
            if entities['Type'] == 'ORGANIZATION' and entities['Score'] >= 0.95:
                entity_text.append(entities['Text'])
```



```
# print(entity_text)
# print('unique entities are ',set(entity_text))
Final_Entity = []
Final_results =athena_connection(set(entity_text),Final_Entity)

#Extracting keyphrase using comprehend
final_sentiment = []
keyphrase_1 = comprehend.detect_sentiment(Text = analysis_string_1,
LanguageCode = "en")
keyphrase_2 = comprehend.detect_sentiment(Text = analysis_string_2,
LanguageCode = "en")

final_sentiment.append(keyphrase_1['Sentiment'])
final_sentiment.append(keyphrase_2['Sentiment'])

result_response1 = "Here are the stocks redditors are talking about.."
result_response2 = " " + str(Final_results)

if final_sentiment[0] == final_sentiment[1]:
    result_response3 = "Sentiment of the overall Reddit Conversation seems to
be: " + str(final_sentiment[0])
else:
    result_response3 = " Sentiment of the overall Reddit Conversation seems to
be: " + "Mixed"

combined_text = result_response1 + result_response2 + result_response3

except Exception as err:
    print(err)

result = {
    "sessionAttributes": {},
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": combined_text
        }
    }
}

return result
```

Once we get entities from comprehend, we need to check if it is valid Entity or not. Since here we are talking about stocks. We specifically want companies/organizations. To validate this we used Nasdaq stock

data which we loaded in athena and checked if the Entity is a valid entity or not. If we found the valid Entity it is returned to the main function. This Lambda function returns the stock names which customer wants to know from the text and overall Sentiment of the text. Overall Sentiment helps to understand the current market situation and can invest in particular stock.

```
# Check if Entity is valid Entity from athena query
```

```
def athena_connection(list, Final_Entity):
    # athena constant
    DATABASE = 'reddit_stock_data'
    TABLE = 'data_dictionary_ticker'
    # S3 constant
    S3_OUTPUT = 's3://output-athena-results/'
    S3_BUCKET = 'output-athena-results'

    # number of retries
    RETRY_COUNT = 20

    COLUMN1, COLUMN2 = 'symbol', 'name'

    for list_each in list :
        # print(list_each)
        # print(type(list_each))
        if list_each.isalpha():
            # created query AND 'symbol' like % % OR 'Name' like % %"
            query = "SELECT count(*) FROM %s.%s where %s like '%s' OR %s like '%s';"
            % (DATABASE, TABLE, COLUMN1, '%' + list_each + '%', COLUMN2, '%' + list_each + '%')
            # print(query)

            # athena client
            client = boto3.client('athena')

            # Execution
            response = client.start_query_execution(
                QueryString=query,
                QueryExecutionContext={
                    'Database': DATABASE
                },
                ResultConfiguration={
                    'OutputLocation': S3_OUTPUT,
                }
            )
```

```
# get query execution id
query_execution_id = response['QueryExecutionId']
# print('query_execution_id is',query_execution_id)

# get execution status
for i in range(1, 1 + RETRY_COUNT):

    # get query execution
    query_status =
client.get_query_execution(QueryExecutionId=query_execution_id)
    query_execution_status =
query_status['QueryExecution']['Status']['State']
    # print(query_execution_status)
    # print(query_status)

    if query_execution_status == 'SUCCEEDED':
        print("STATUS:" + query_execution_status)
        results =
client.get_query_results(QueryExecutionId=query_execution_id)
        final_result = (results['ResultSet']['Rows'])
        final_exe = int(final_result[1]['Data'][0]['VarCharValue'])
        # print(final_exe)
        # print(results)
        #print(type(results))
        if (final_exe) >= 1:
            Final_Entity.append(str(list_each))
            #print('Final Entity is',Final_Entity)
            # print(list_each)
            break

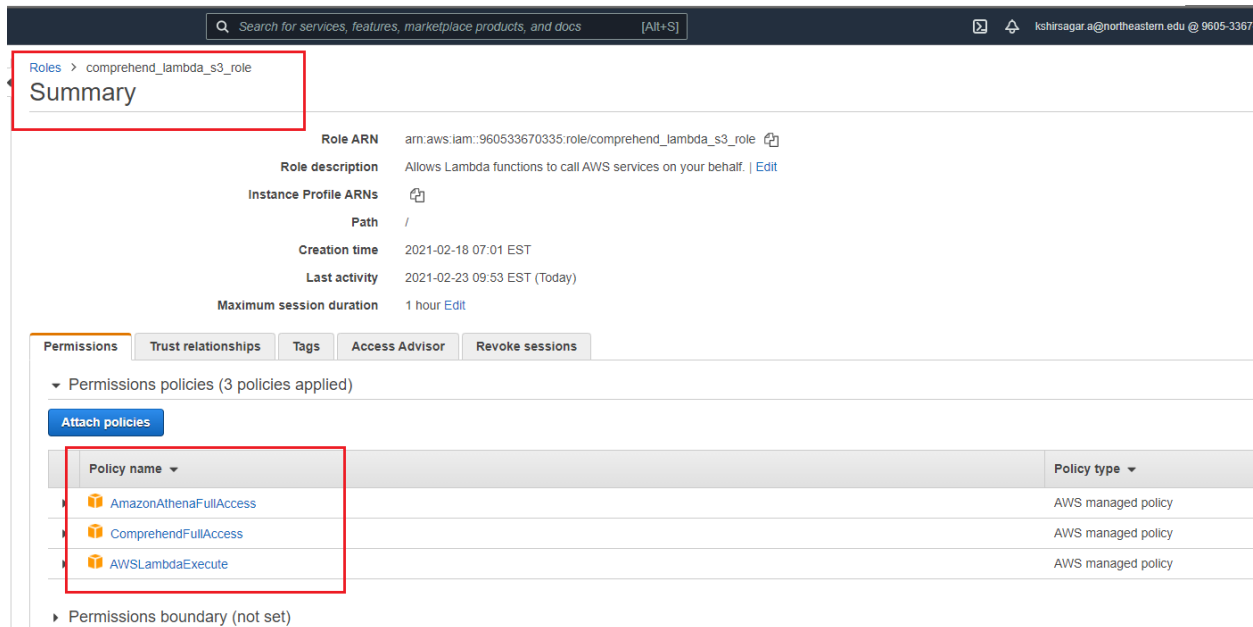
        if query_execution_status == 'FAILED':
            raise Exception("STATUS:" + query_execution_status)

        else:
            print("STATUS:" + query_execution_status)
            time.sleep(i)

# else:
#     client.stop_query_execution(QueryExecutionId=query_execution_id)
#     raise Exception('TIME OVER')
# break

return Final_Entity
```

## IAM Roles and Policy associated with amazon Comprehend



The screenshot shows the AWS IAM console interface for the role `comprehend_lambda_s3_role`. The **Summary** tab is selected, displaying the following details:

- Role ARN:** `arn:aws:iam::960533670335:role/comprehend_lambda_s3_role`
- Role description:** Allows Lambda functions to call AWS services on your behalf.
- Instance Profile ARNs:** (None listed)
- Path:** `/`
- Creation time:** 2021-02-18 07:01 EST
- Last activity:** 2021-02-23 09:53 EST (Today)
- Maximum session duration:** 1 hour

Under the **Permissions** tab, it shows that 3 policies are applied:

Policy name	Policy type
AmazonAthenaFullAccess	AWS managed policy
ComprehendFullAccess	AWS managed policy
AWSLambdaExecute	AWS managed policy

### 6.4.Athena and Glue

We are using AWS Glue to create a data catalog by importing the Nasdaq stock market data from the S3 bucket. AWS Glue allows us to detect the schema of our data and then create the database by automatically detecting the schema and populating it. We created our database `reddit_stock_data` with the table `data_dictionary_ticker`, which is then used by the Athena query engine which is invoked by a lambda function that is using a data dictionary to compare and validate the entities (stock names) that are discussed in the Reddit forums with the `data_dictionary_ticker` table in our database which consists of stock data from the market so that we are only accepting valid stock names from the Reddit discussions and not any other entity which is not a stock name, but some other organization name. Thus, by using the query service of Athena, we are able to filter the valid stock names and return the results to both the chat interface and the QuickSight dashboard.

## Reddit Stock Analysis using AWS Cloud Analytics Services

The screenshot displays two AWS management console interfaces. The top interface is the AWS Glue 'Tables' page, showing a table named 'data\_dictionary\_ticker' in the 'reddit\_stock\_data' database. The bottom interface is the AWS Athena 'Query editor', where the same database and table are selected as the data source. A SQL query is entered: 'SELECT \* FROM "reddit\_stock\_data"."data\_dictionary\_ticker" limit 10;'. The query is executed, and the results are displayed below.

**AWS Glue Tables**

Name	Database	Location	Classification	Last updated
data_dictionary_ticker	reddit_stock_data	s3://data-dictionary-ticker-names/	csv	20 February 2021 5:55 AM UTC-5
elb_logs	sampledb	s3://athena-examples-us-east-1/e...	Unknown	19 February 2021 5:14 AM UTC-5

**AWS Athena Query editor**

**Data source:** AwsDataCatalog

**Database:** reddit\_stock\_data

**Tables (1):** data\_dictionary\_ticker

**Query:** 1. SELECT \* FROM "reddit\_stock\_data"."data\_dictionary\_ticker" limit 10;

**Run query** (Run time: 0.48 seconds, Data scanned: 2.11 KB)

**Results:**

## IAM Roles and Policy associated with Athena and Glue

The screenshot shows the AWS IAM console 'Summary' page for the 'Lambda-Athena-FullAccess' role. The role is associated with the 'AmazonS3FullAccess' and 'AmazonAthenaFullAccess' policies. The 'Permissions' tab is selected, showing the list of policies applied to the role.

**Role Summary:**

- Role ARN:** arn:aws:iam::960533670335:role/Lambda-Athena-FullAccess
- Role description:** Allows Lambda functions to call AWS services on your behalf.
- Instance Profile ARNs:** /
- Path:** /
- Creation time:** 2021-02-19 10:56 EST
- Last activity:** 2021-02-23 02:28 EST (Today)
- Maximum session duration:** 1 hour

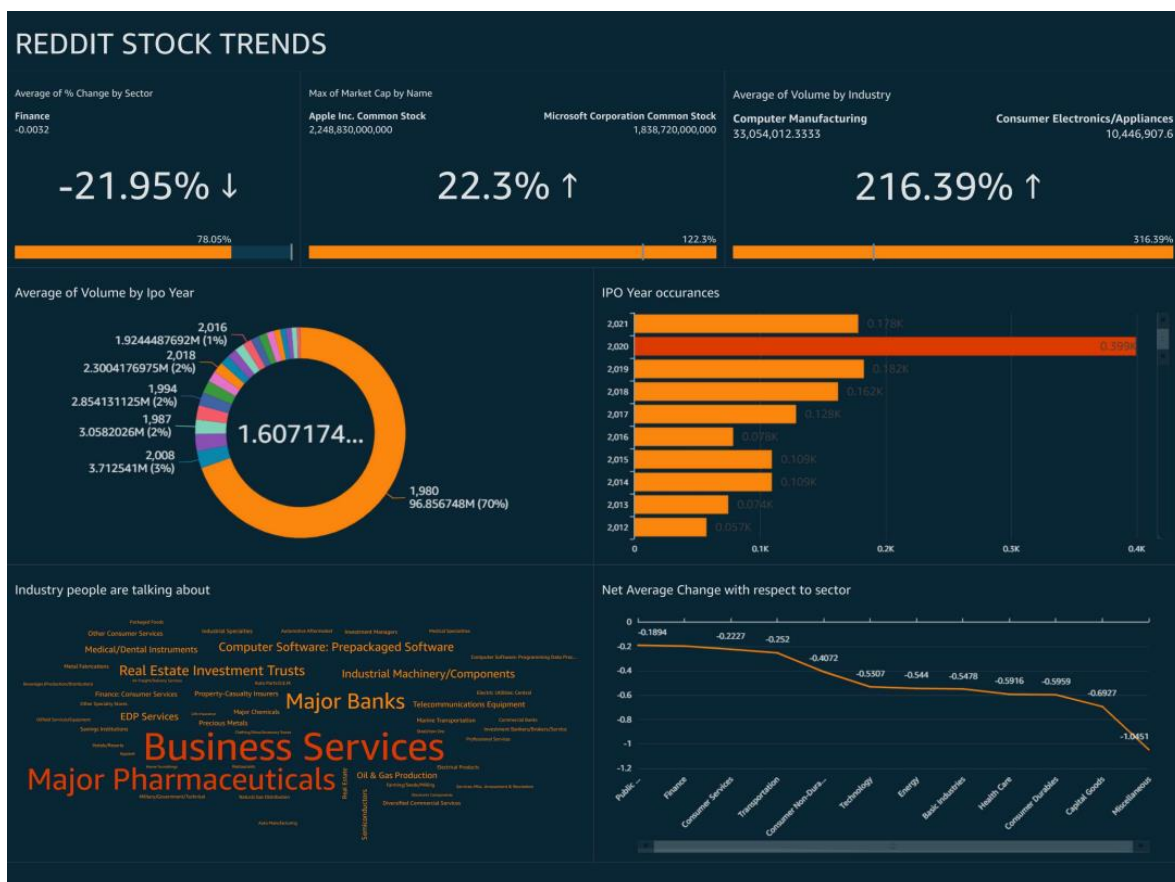
**Permissions:**

- Permissions policies (2 policies applied)
- Attach policies
- Policy name: AmazonS3FullAccess (AWS managed policy)
- Policy name: AmazonAthenaFullAccess (AWS managed policy)
- Permissions boundary (not set)

## 6.5. Quicksight

The data was collected to Quicksight using Athena which was loaded manually. The dashboard was made with various graphs like Donut chart, Bar graph, word cloud, line graph, etc. The dashboard indicated three key performance indices which were Average percentage change by a specific sector, Maximum market cap by name and Average volume per industry. Some interesting facts which we got from the visuals were, the industries which occurred a greater number of times were Business Services and Pharmaceuticals and the net percentage change was the highest for the miscellaneous sector. Filters were attached to the dashboard which made it more interactive. Using this filter information related to a specific symbol or a name was easy to visualize.

### Dashboard:



## IAM Roles and Policy associated with Quicksight

Search for services, features, marketplace products, and docs

[Alt+S]

kshirsagar.a@northeastern.edu

Roles > aws-quicksight-service-role-v0

Summary

Role ARN

am:aws:iam::960533670335:role/service-role/aws-quicksight-service-role-v0

Role description

[Edit](#)

Instance Profile ARNs

Path

/service-role/

Creation time

2021-02-17 09:16 EST

Last activity

2021-02-20 17:49 EST (2 days ago)

Maximum session duration

1 hour [Edit](#)

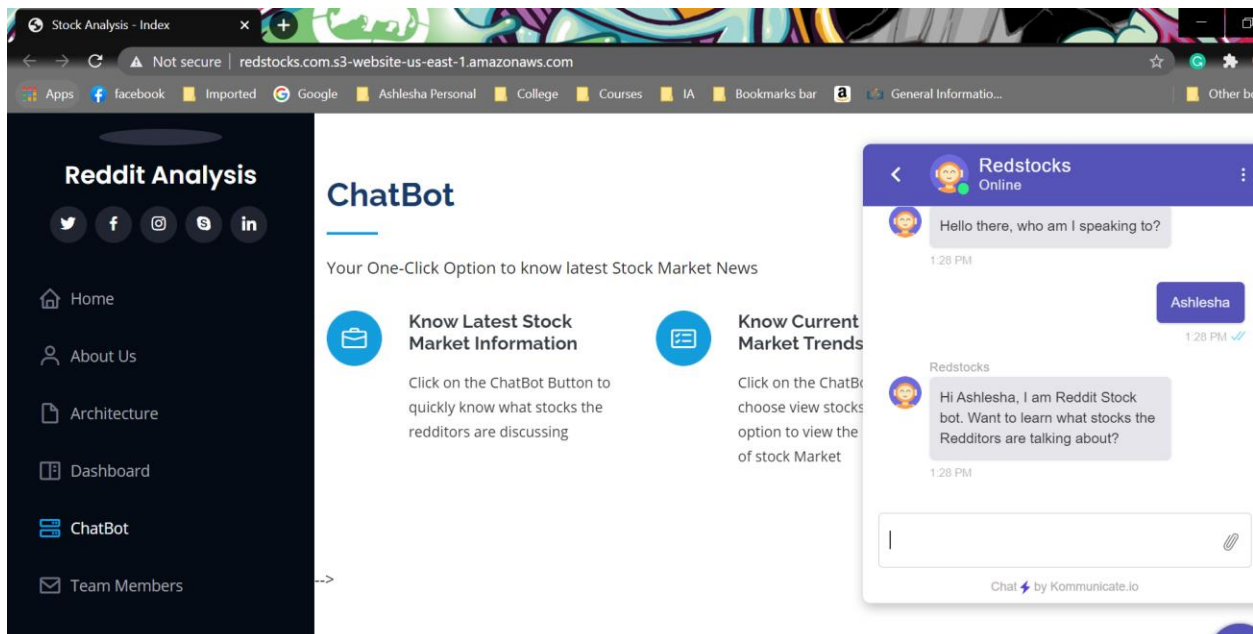
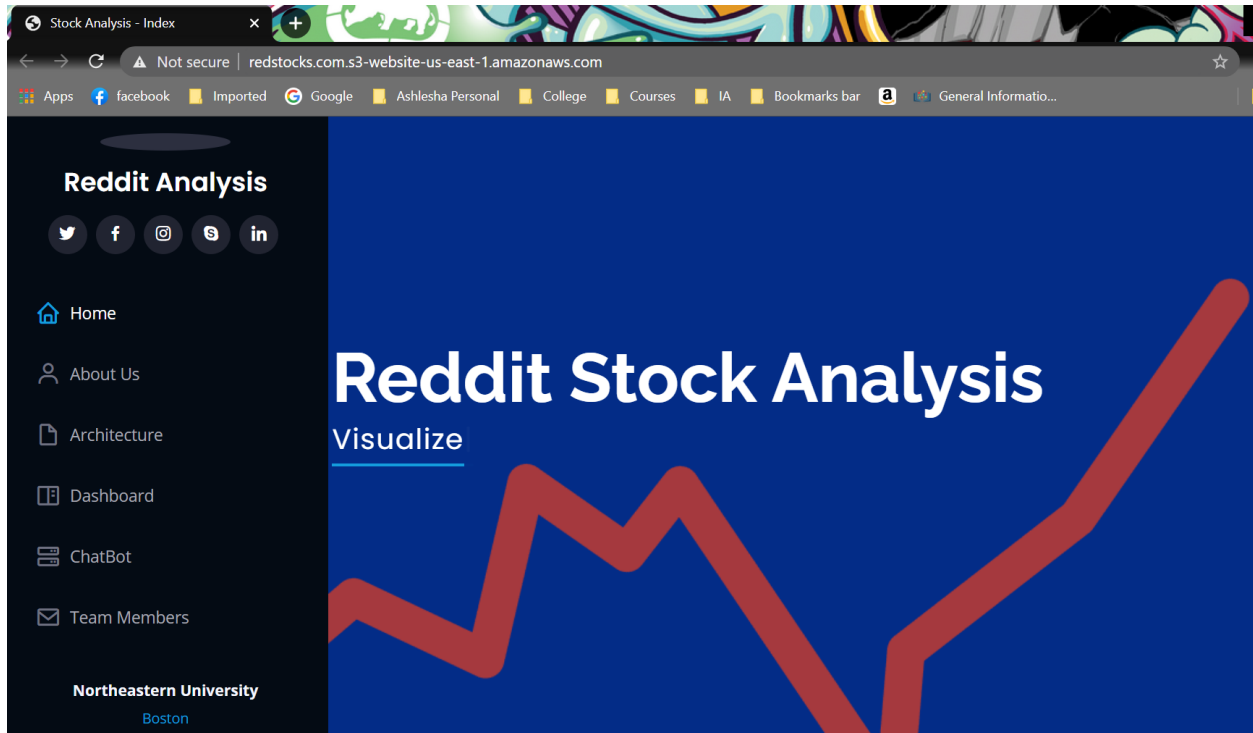
PermissionsTrust relationshipsTagsAccess AdvisorRevoke sessions

▼ Permissions policies (7 policies applied)

Attach policies

Policy name ▼	Policy type ▼
▶ <a href="#">AWSQuickSightIAMPolicy</a>	Managed policy
▶ <a href="#">AWSQuickSightS3Policy</a>	Managed policy
▶ <a href="#">AWSQuickSightRedshiftPolicy</a>	Managed policy
▶ <a href="#">AWSQuickSightRDSPolicy</a>	Managed policy
▶ <a href="#">AWSQuickSightAthenaAccess</a>	AWS managed policy
▶ <a href="#">QuickSightAccessForS3StorageManagementAnalyticsReadOnly</a>	AWS managed policy
▶ <a href="#">AWSQuickSightToAnalyticsAccess</a>	AWS managed policy

## 7.Web interface





## **8.Challenges**

### **1.Limitations in the size of data that can analyze at a time using comprehend:**

AWS comprehend was used to process the data and determine the entities, key phrases and the sentiments of various Reddit conversations. The primary challenge we encountered here was to process large amounts of data using AWS Comprehend. AWS Comprehend can only process 5kb of data at a time [3] and we overcame this by dividing the data extracted from reddit into lists so as to enable Comprehend to accommodate no more than 5kb of data for processing. The results after the analysis were merged back to suit our requirements.

### **2.Limitations on the output size of the lambda function used in fulfilling an intent:**

AWS lex has a limitation of 16kb [4] on the size of the data object that can be returned by a lambda function when used in response to an intent. As the objective of the project was to formulate a conversational informative chatbot, it was important to overcome this challenge so as to provide an effective insight to the end user regarding the on goings of the stock market. We overcame this challenge by filtering the redundant details in our output message based on the confidence levels of the entities and sentiments obtained as a result of data processing via AWS Comprehend. This enabled us to provide an effective and a concise statement to the end user.

### **3.Unavailability of real-time data streaming API for stock market data:**

In order to be able to provide easy access to the end users to stock market data in the form of a dashboard, we were to access stock market data in real time and due to the unavailability of any API that will enable us fulfill this, we performed ETL on data obtained from Nasdaq in order to fuel our quick sight dashboard.

## **9.Future Scope**

In the future versions of this Lex chatbot, we can incorporate more intents where the user is presented with an option to mention specific names of stocks, whose trends they want to analyze. The Lex bot would then return the results from the text analysis of Reddit discussions for those specific stocks, which would alert the user of any upcoming activity by the online traders collaborating on Reddit for those specific stocks. The user would also get to see a custom dashboard for those particular stocks and the trend analysis of how they performed in the market in the recent days that would give sufficient data to the user to make an informed decision while trading in the market.

## **10.Conclusion**

To conclude, a website can be deployed for assisting the stock market traders with making an informed decision, where they can read all the necessary information on a single webpage, especially if they are new to trading. The chatbot would ask them for specific requirements and provide them with detailed information, be it a Reddit discussion analysis or a current stock market trend analysis. This will give the users with all the valuable information at a single site in a user-friendly manner, using which they can confidently start trading without fearing any unaccepted market move by people collaborating on Reddit.

## **11.References**

- [1] Phillips, Lorenz, (Jan 27, 2021). GameStop vs. Wall Street. Retrieved from **<https://www.nytimes.com/2021/01/27/business/gamestop-wall-street-bets.html>**
  
- [2] AWS (n.d.). Featured Services. Retrieved from **<https://aws.amazon.com/#>**
  
- [3] AWS Comprehend: Guidelines and Quotas  
**<https://docs.aws.amazon.com/comprehend/latest/dg/guidelines-and-limits.html>**