CPSC 471 Database Management Systems

Initial Functional Design

Group Number: *21*

Laura Timm

30064647

Abhay Khosla

30085789

Hassan Khan

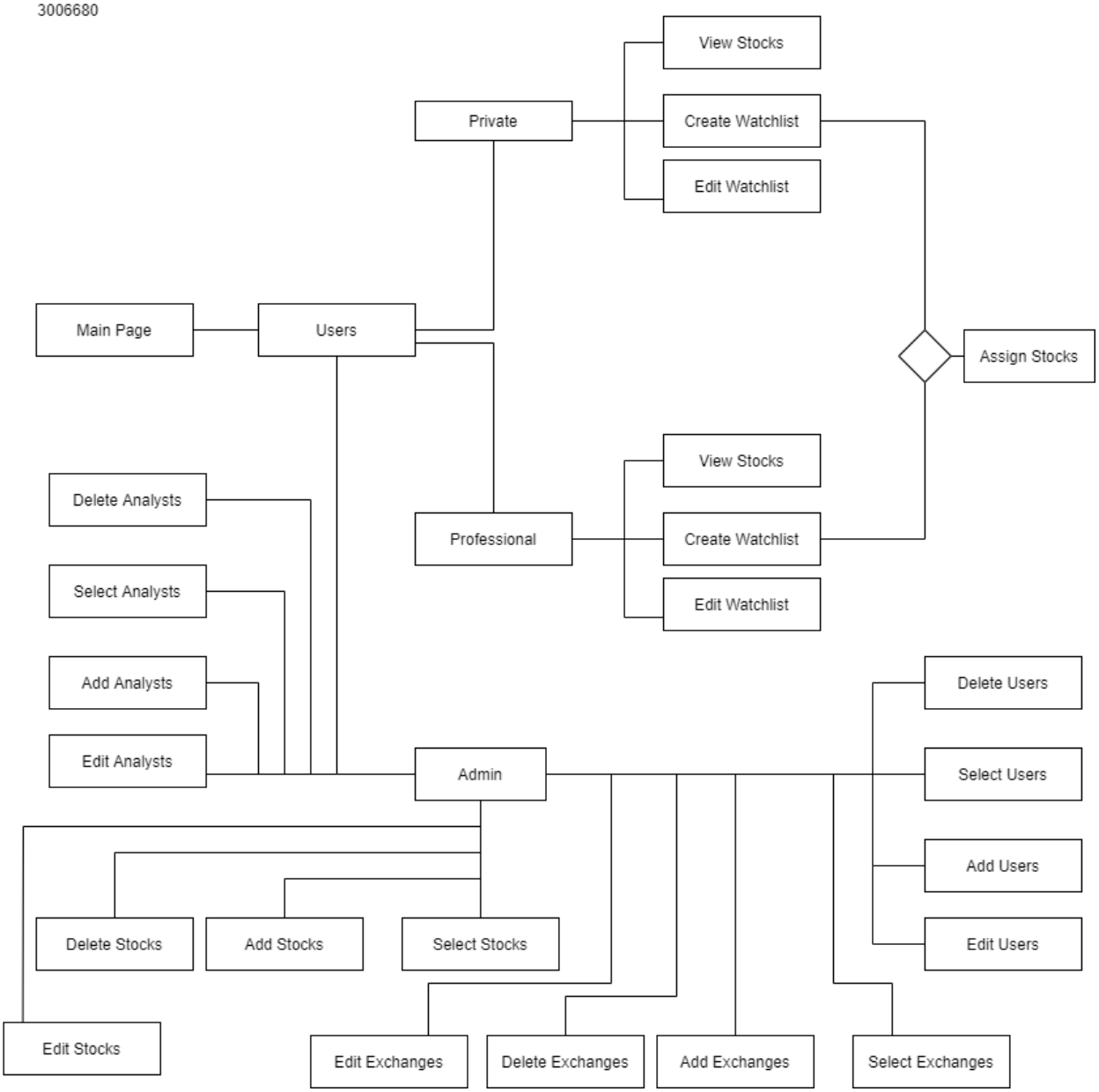
3006680

TABLE OF CONTENTS

HIPO Diagram:

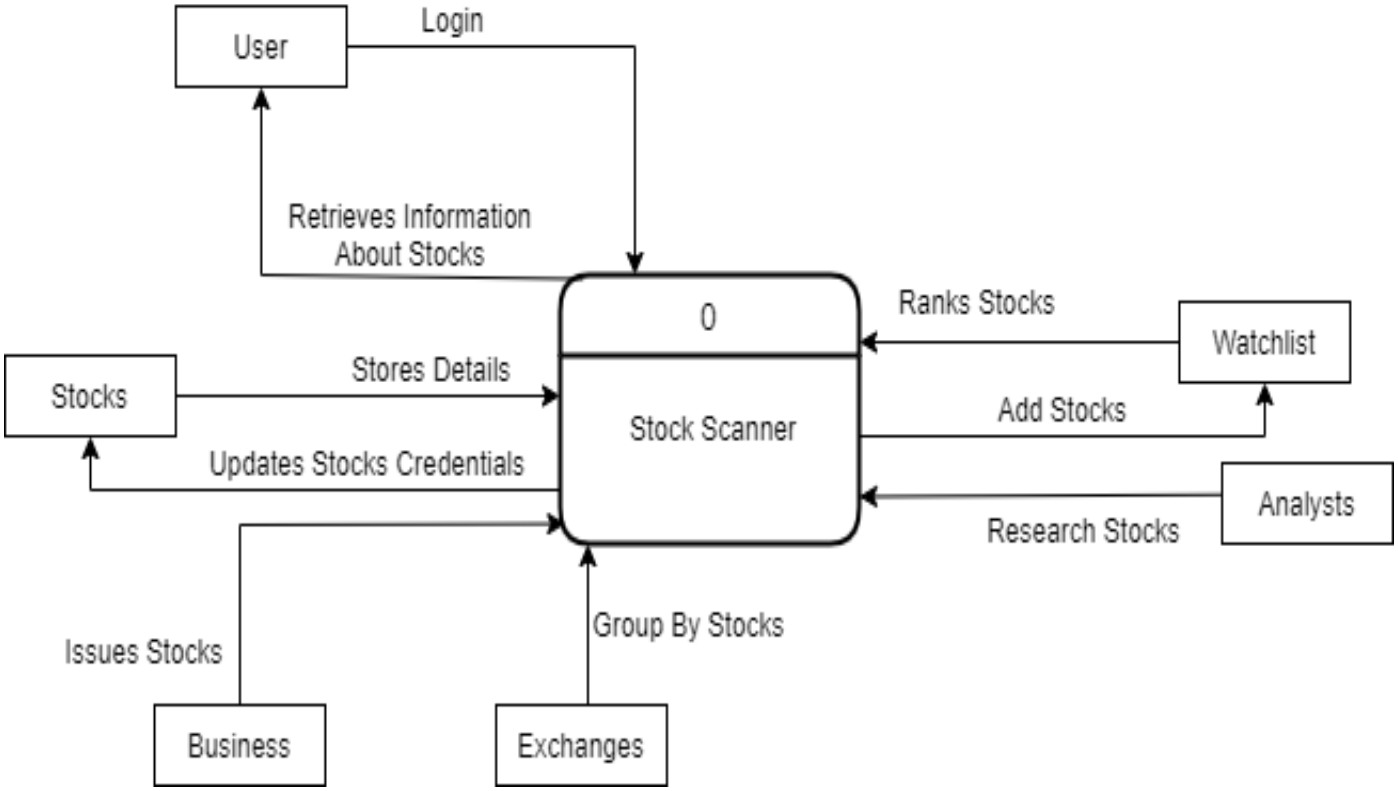
Group Number: 21

Names:
Laura Timm
30064647
Abhay Khosla
30085789
Hassan Khan
3006680



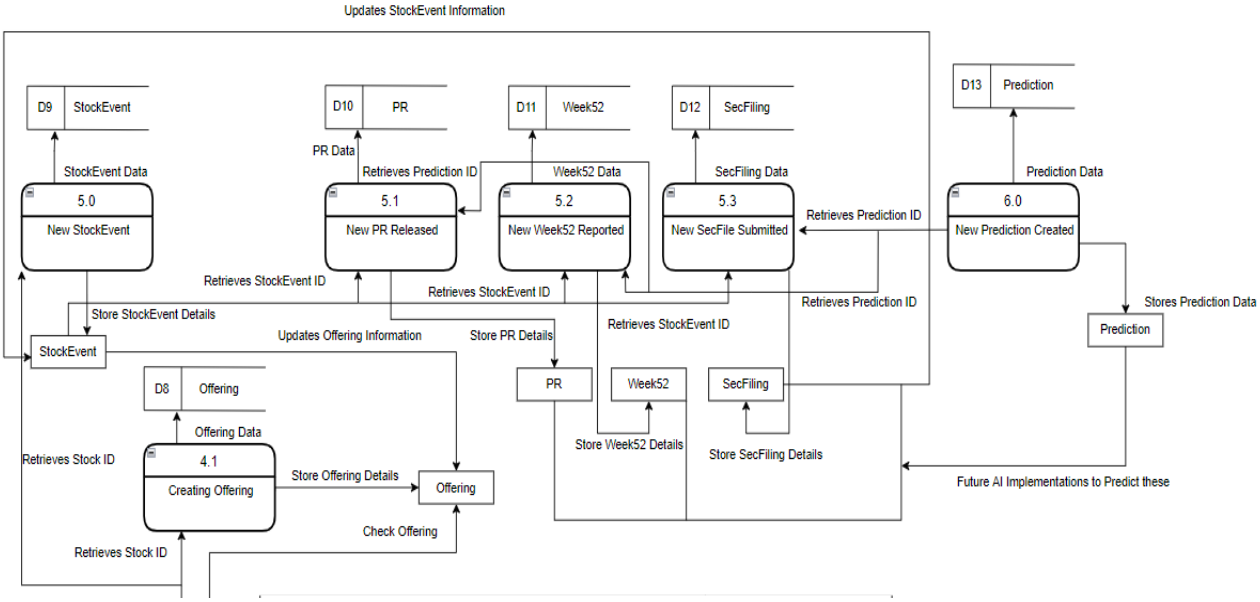
Not including the input-process-output diagram since it's optional

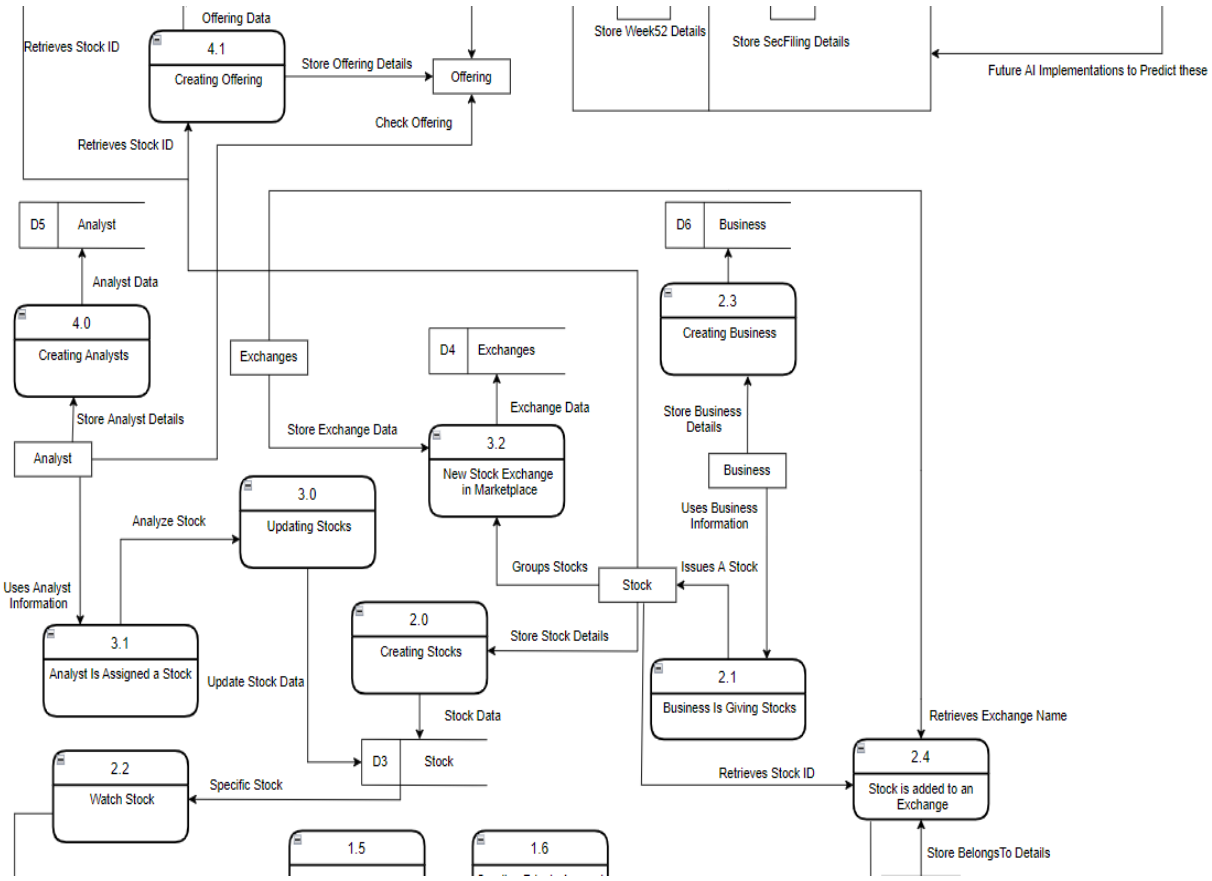
Context Diagram:

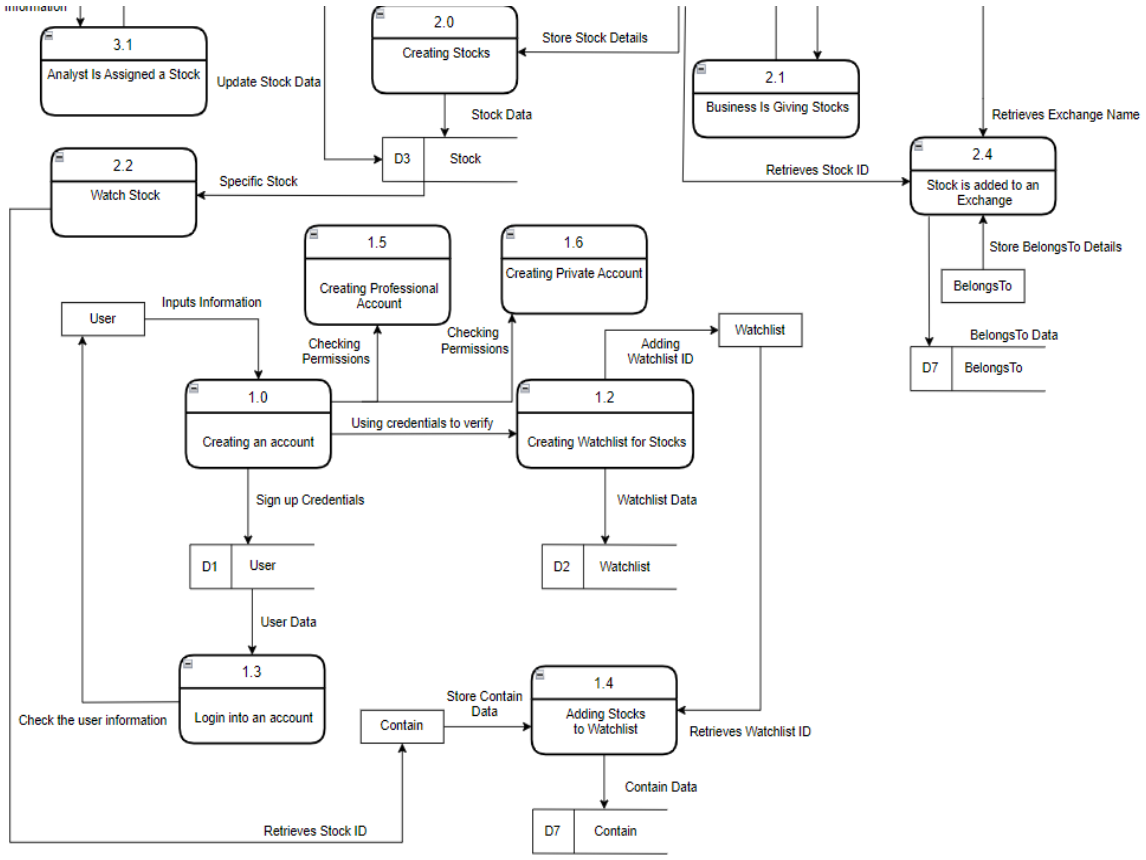


Data Flow Diagram:

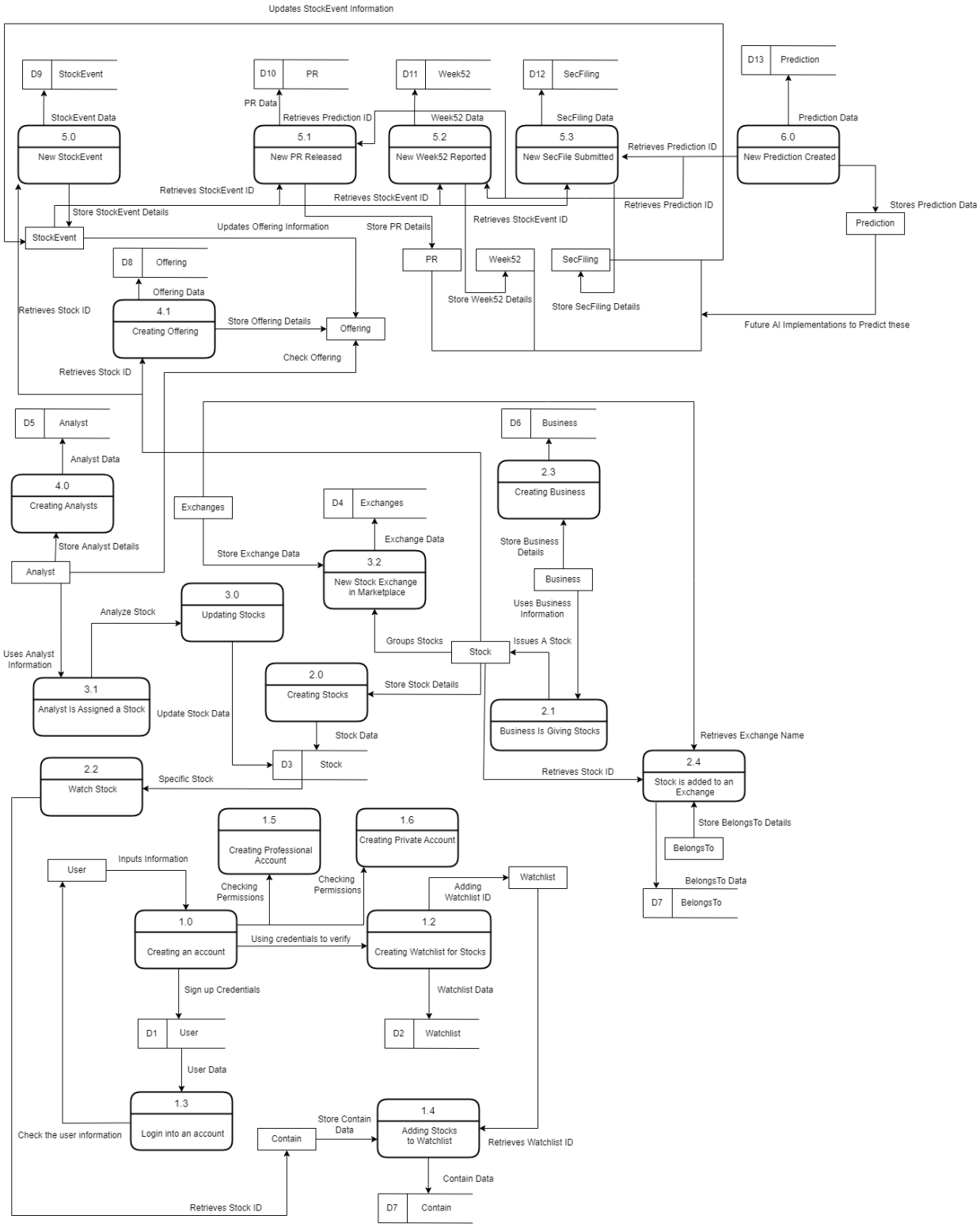
Broken Down components for better viewing:




Offering Data
Retrieves Stock ID
4.1
Creating Offering
Store Offering Details
Offering
Check Offering
Retrieves Stock ID
Store Week52 Details
Store SecFiling Details
Future AI Implementations to Predict these
D5
Analyst
Analyst Data
4.0
Creating Analysts
Store Analyst Details
Analyst
Exchanges
D4
Exchanges
Exchange Data
Store Exchange Data
3.2
New Stock Exchange in Marketplace
D6
Business
2.3
Creating Business
Store Business Details
Business
Uses Business Information
3.0
Updating Stocks
Analyze Stock
Groups Stocks
Stock
Issues A Stock
Uses Analyst Information
3.1
Analyst Is Assigned a Stock
Update Stock Data
2.0
Creating Stocks
Store Stock Details
2.1
Business Is Giving Stocks
Stock Data
D3
Stock
Retrieves Exchange Name
2.2
Watch Stock
Specific Stock
Retrieves Stock ID
2.4
Stock is added to an Exchange
Store BelongsTo Details
1.5
1.6

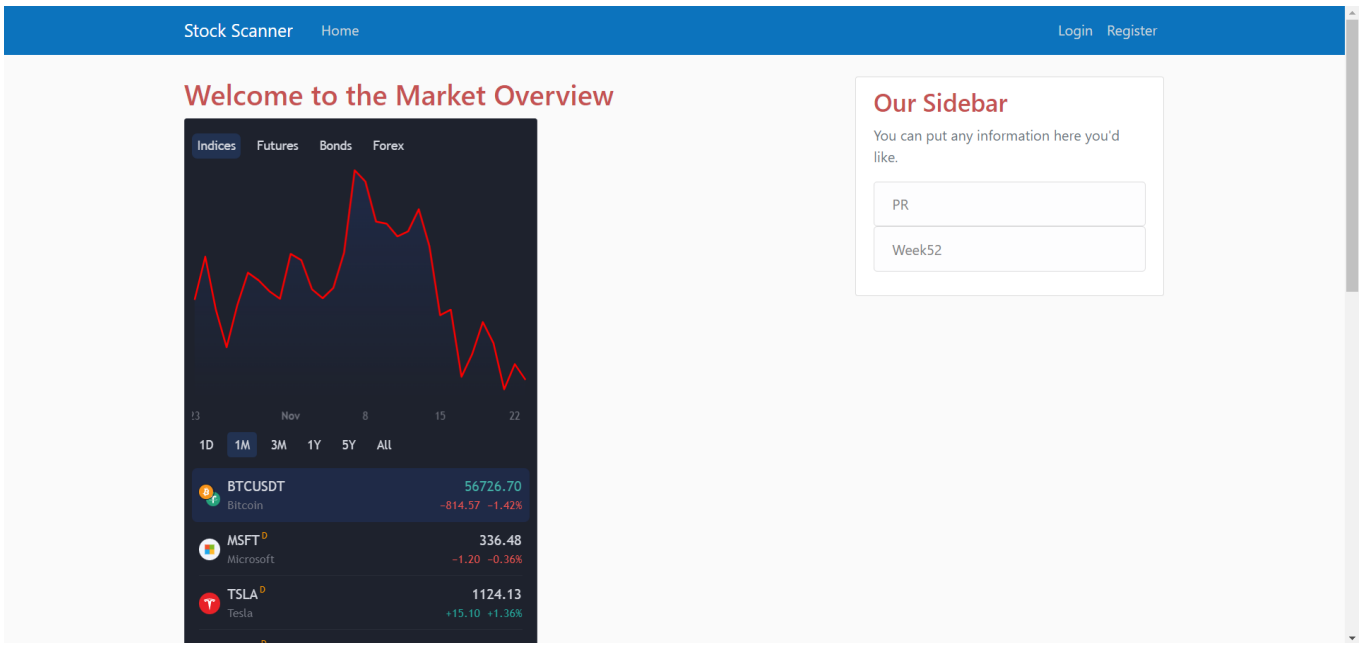

3.1
Analyst Is Assigned a Stock
Update Stock Data
2.0
Creating Stocks
Store Stock Details
Stock Data
D3
Stock
2.1
Business Is Giving Stocks
Retrieves Exchange Name
2.2
Watch Stock
Specific Stock
Retrieves Stock ID
2.4
Stock is added to an Exchange
Store BelongsTo Details
BelongsTo
BelongsTo Data
D7
BelongsTo
1.5
Creating Professional Account
1.6
Creating Private Account
User
Inputs Information
Checking Permissions
Checking Permissions
Adding Watchlist ID
Watchlist
1.0
Creating an account
Using credentials to verify
1.2
Creating Watchlist for Stocks
Sign up Credentials
Watchlist Data
D1
User
D2
Watchlist
User Data
1.3
Login into an account
Check the user information
Store Contain Data
Contain
1.4
Adding Stocks to Watchlist
Retrieves Watchlist ID
Contain Data
D7
Contain
Retrieves Stock ID

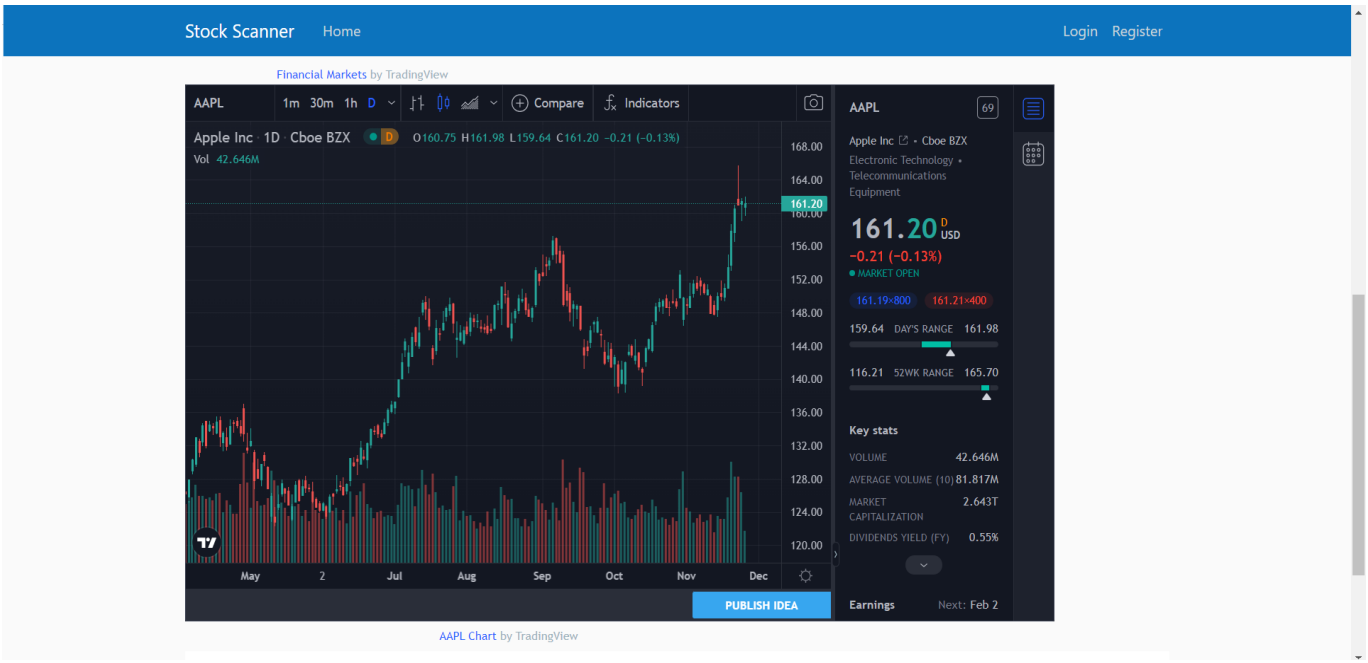
One Image:

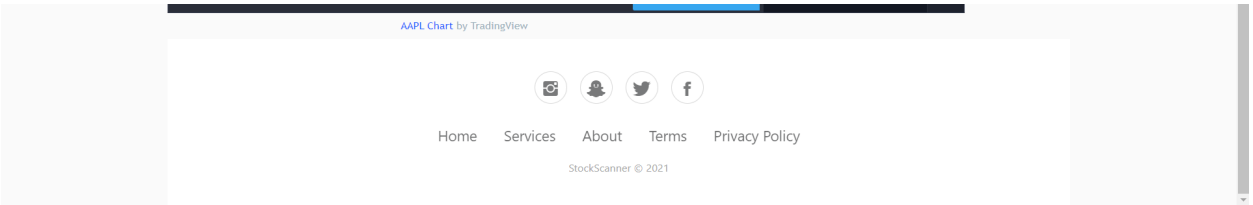

Updates StockEvent Information
D9 StockEvent
D10 PR
D11 Week52
D12 SecFiling
D13 Prediction
StockEvent Data
PR Data
Retrieves Prediction ID
Week52 Data
SecFiling Data
Prediction Data
5.0 New StockEvent
5.1 New PR Released
5.2 New Week52 Reported
5.3 New SecFile Submitted
6.0 New Prediction Created
Retrieves Prediction ID
Retrieves StockEvent ID
Retrieves StockEvent ID
Retrieves Prediction ID
Stores Prediction Data
Store StockEvent Details
Updates Offering Information
Store PR Details
Retrieves StockEvent ID
Prediction
StockEvent
D8 Offering
PR
Week52
SecFiling
Offering Data
Store Week52 Details
Store SecFiling Details
Retrieves Stock ID
4.1 Creating Offering
Store Offering Details
Offering
Future AI Implementations to Predict these
Check Offering
Retrieves Stock ID
D5 Analyst
D6 Business
Analyst Data
2.3 Creating Business
4.0 Creating Analysts
Exchanges
D4 Exchanges
Exchange Data
Store Business Details
Store Analyst Details
Store Exchange Data
3.2 New Stock Exchange in Marketplace
Business
Analyst
3.0 Updating Stocks
Analyze Stock
Uses Business Information
Groups Stocks
Stock
Issues A Stock
Uses Analyst Information
2.0 Creating Stocks
Store Stock Details
3.1 Analyst Is Assigned a Stock
Update Stock Data
2.1 Business Is Giving Stocks
Stock Data
Retrieves Exchange Name
D3 Stock
2.2 Watch Stock
Specific Stock
Retrieves Stock ID
2.4 Stock is added to an Exchange
1.5 Creating Professional Account
1.6 Creating Private Account
Store BelongsTo Details
BelongsTo
User
Inputs Information
Checking Permissions
Checking Permissions
Adding Watchlist ID
Watchlist
BelongsTo Data
D7 BelongsTo
1.0 Creating an account
Using credentials to verify
1.2 Creating Watchlist for Stocks
Sign up Credentials
Watchlist Data
D1 User
D2 Watchlist
User Data
1.3 Login into an account
Check the user information
Contain
Store Contain Data
1.4 Adding Stocks to Watchlist
Retrieves Watchlist ID
Contain Data
D7 Contain
Retrieves Stock ID

Dummy Website:

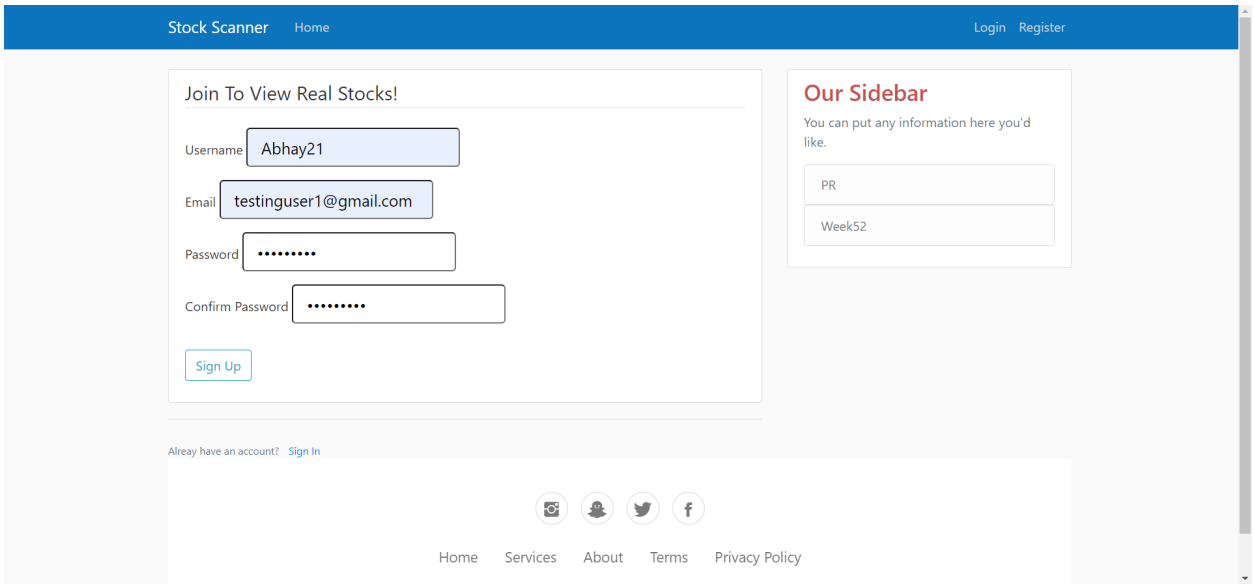
Landing(Home) Page:



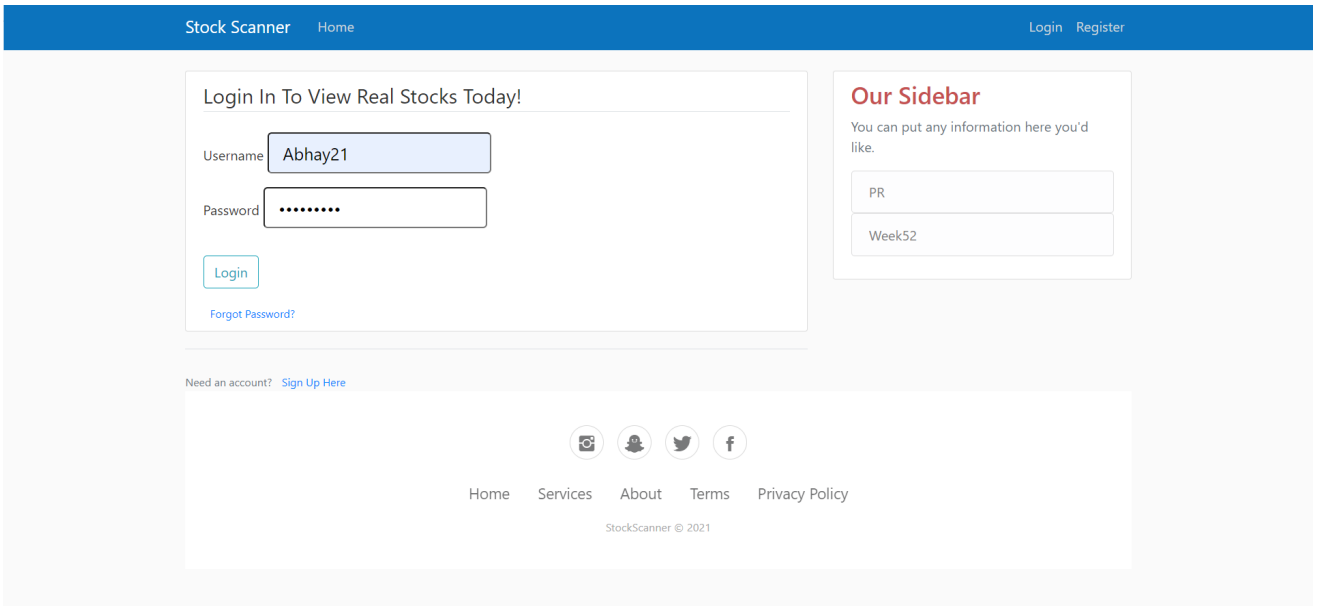




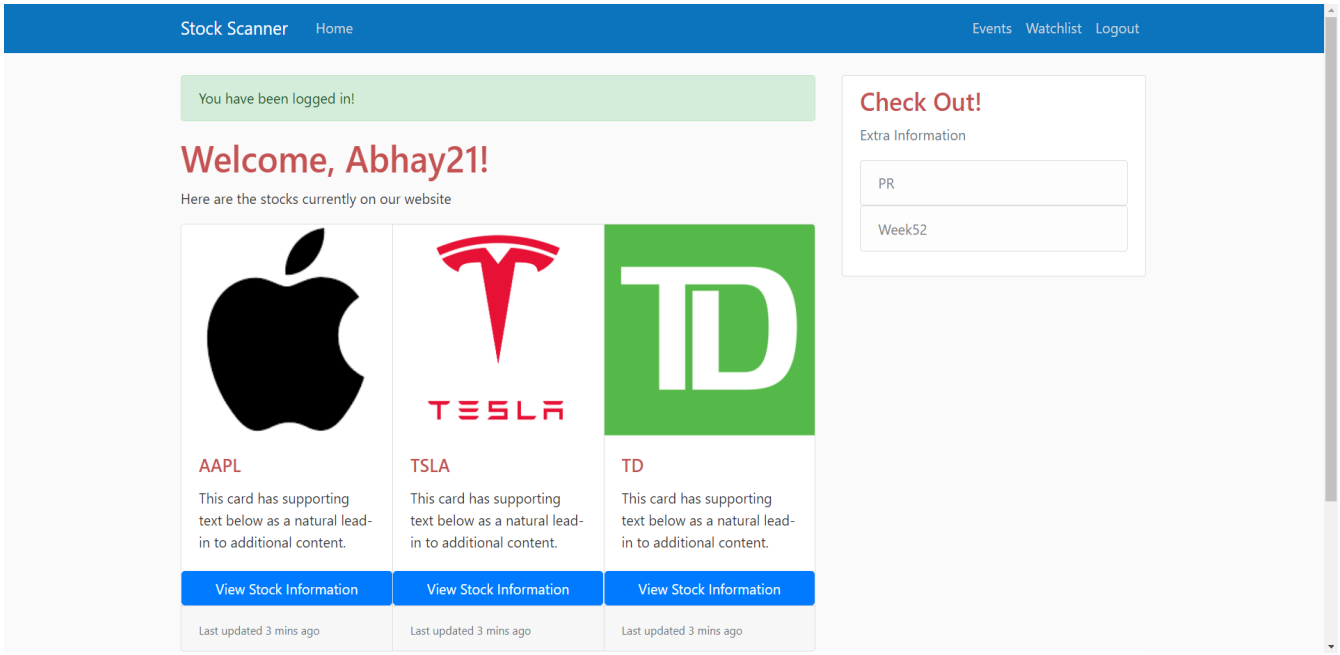
Register Page:



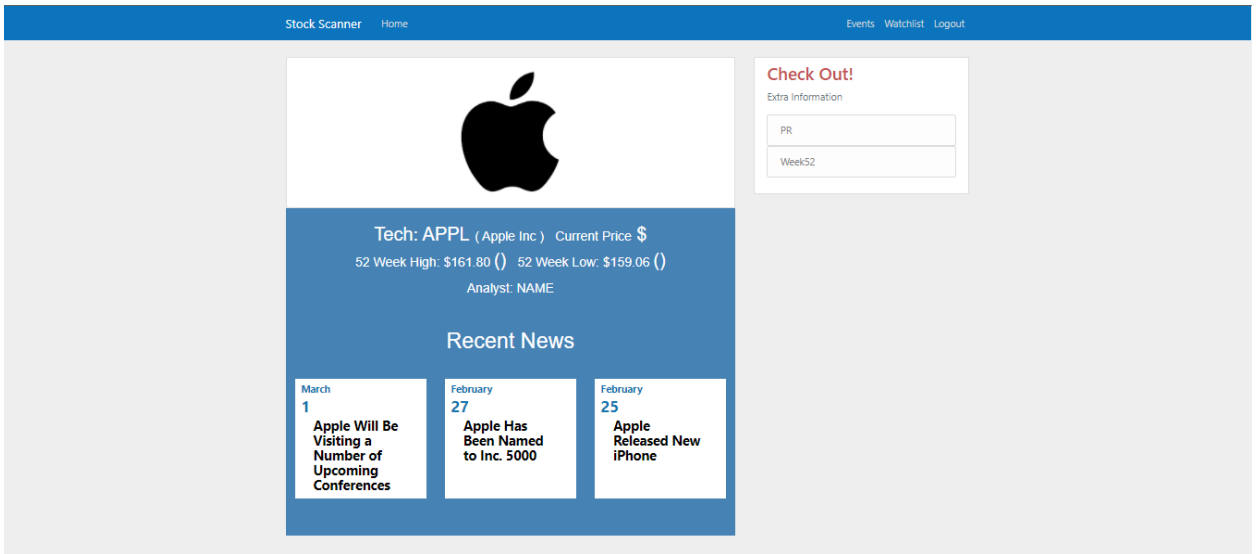
Login Page:



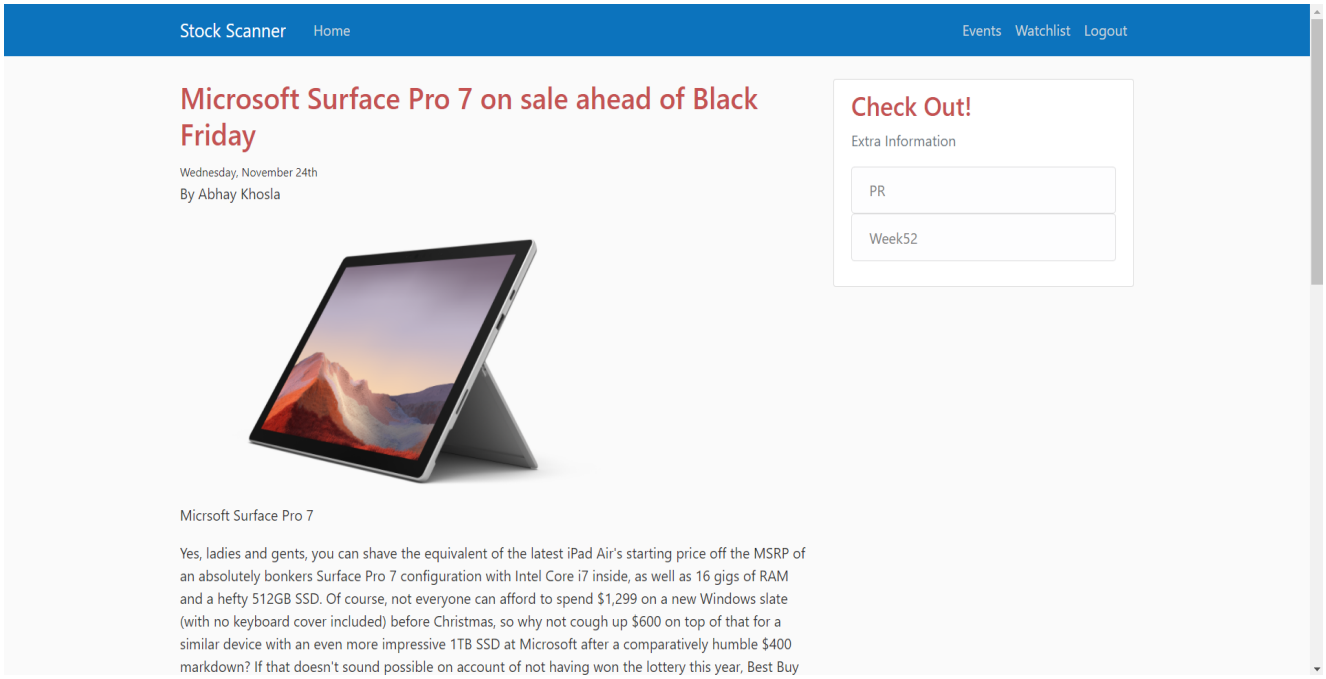
After Logging In:

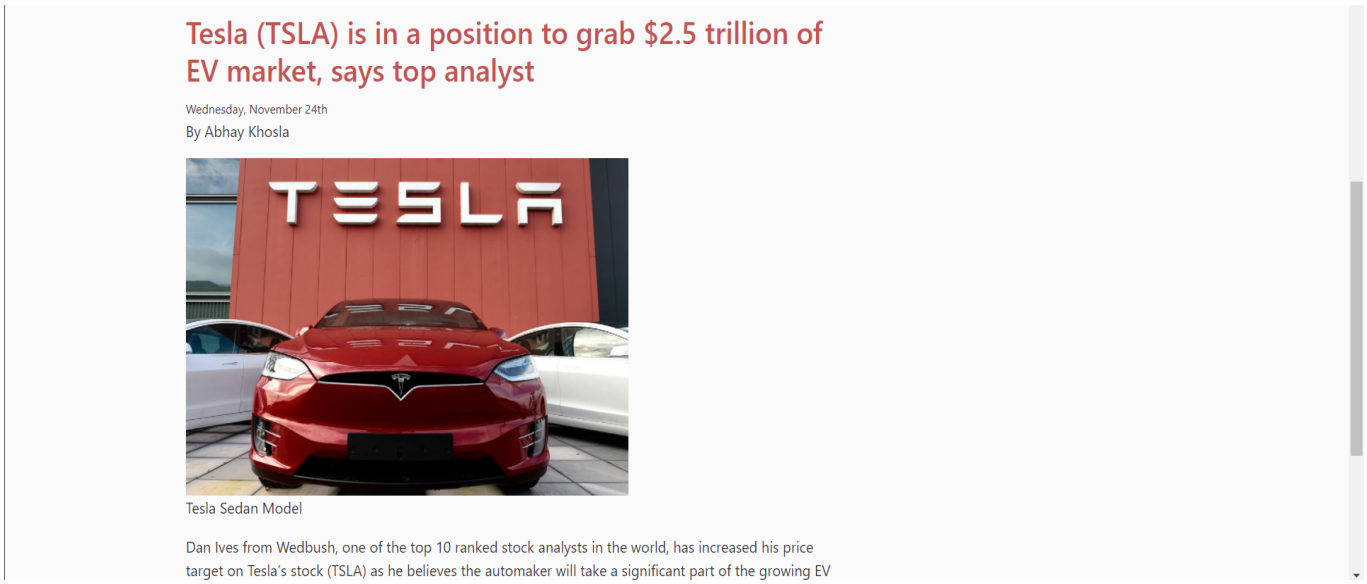


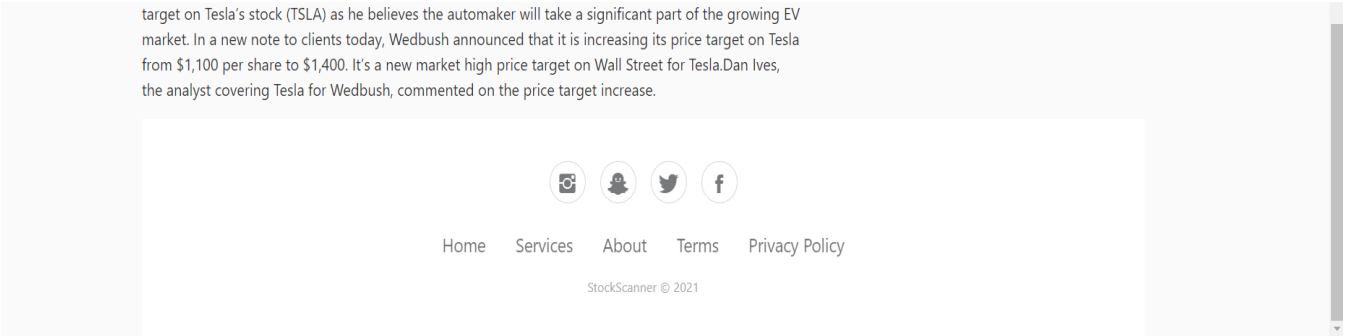
Once you click on Stock Information:



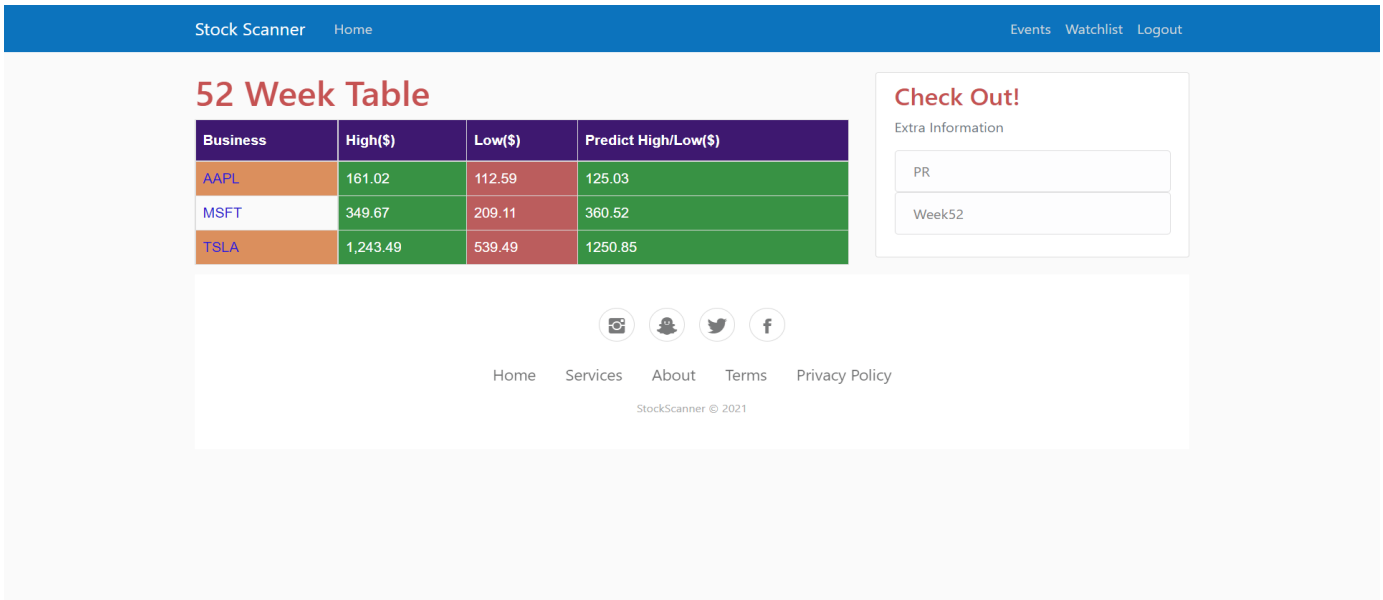
PR(News/Headlines of the Stocks) Page:



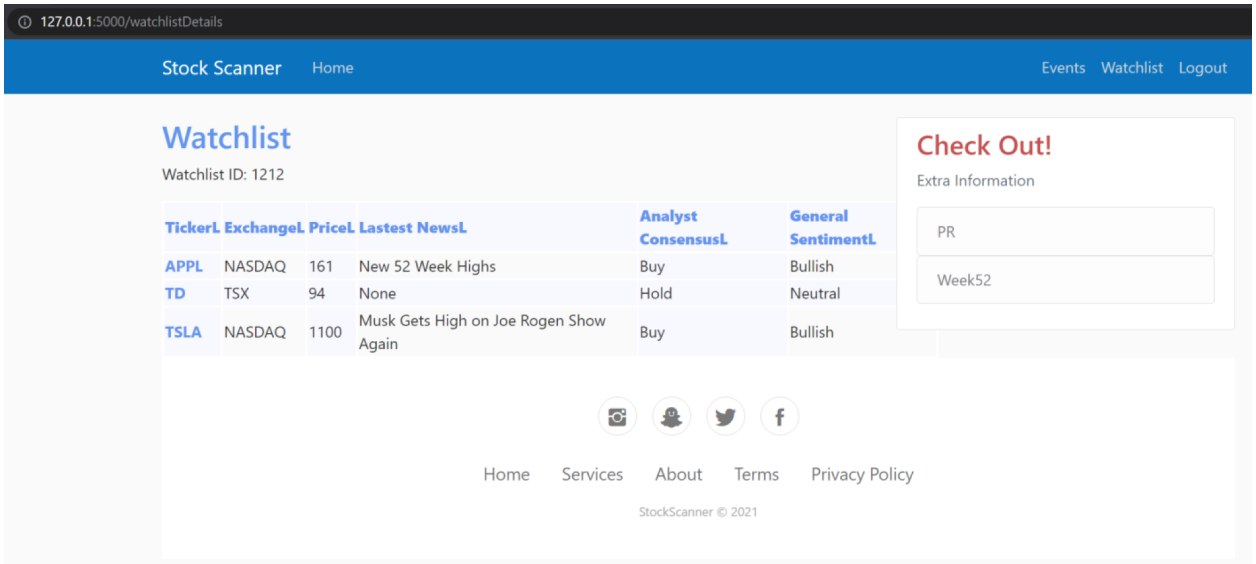



target on Tesla's stock (TSLA) as he believes the automaker will take a significant part of the growing EV market. In a new note to clients today, Wedbush announced that it is increasing its price target on Tesla from $1,100 per share to $1,400. It's a new market high price target on Wall Street for Tesla.Dan Ives, the analyst covering Tesla for Wedbush, commented on the price target increase.
Home
Services
About
Terms
Privacy Policy
StockScanner © 2021

Week52Page:

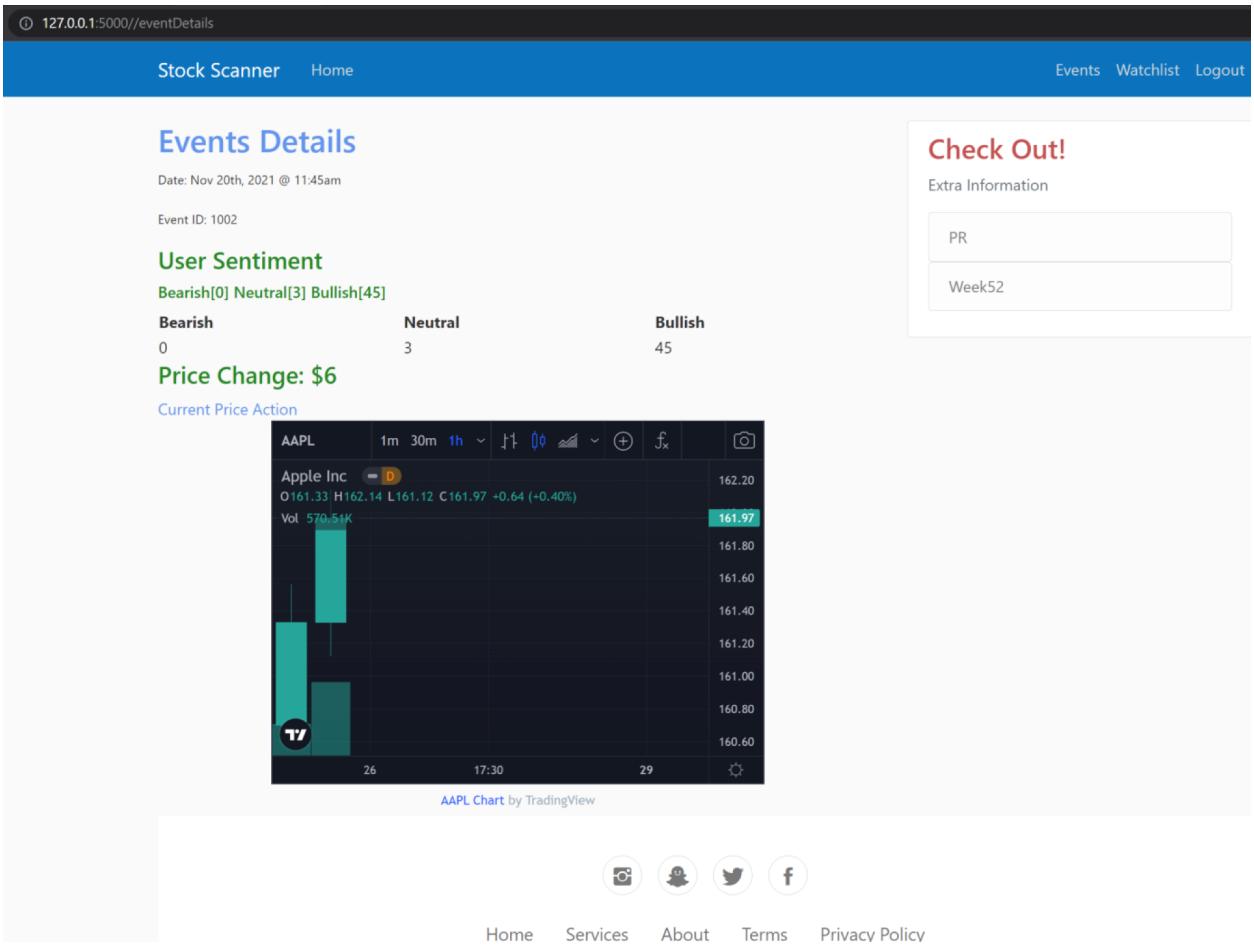

Stock Scanner
Home
Events
Watchlist
Logout
52 Week Table
Business | High($) | Low($) | Predict High/Low($)
AAPL | 161.02 | 112.59 | 125.03
MSFT | 349.67 | 209.11 | 360.52
TSLA | 1,243.49 | 539.49 | 1250.85
Check Out!
Extra Information
PR
Week52
Home
Services
About
Terms
Privacy Policy
StockScanner © 2021

Watchlist:



Ticker	Exchange	Price	Lastest News	Analyst Consensus	General Sentiment
APPL	NASDAQ	161	New 52 Week Highs	Buy	Bullish
TD	TSX	94	None	Hold	Neutral
TSLA	NASDAQ	1100	Musk Gets High on Joe Rogen Show Again	Buy	Bullish

Event Details:



API Blueprint:

Endpoint 1:

Description: Creating new Business.
URL: http://127.0.0.1:5000/business
Method: POST
Input(JSON format):

```
{
   "Business_ID": "TD",
   "Address": "1234 Toronto",
   "Founding_Date": "1855-01-01",
   "Business_Name": "TD Canada Trust"
}
```

Output:"Business updated successfully"

Endpoint 2:

Description: Get all Businesses.
URL: http://127.0.0.1:5000/business
Method: GET
Input: None
Output(JSON format):

```
{
  "Business": [
    [
      "AAPL",
      "1234 California Road",
      "Sun, 04 Jan 1976 00:00:00 GMT",
      "Apple"
    ],
    [
      "TD",
      "1234 Toronto",
      "Mon, 01 Jan 1855 00:00:00 GMT",
      "TD Canada Trust"
    ]
  ]
}
```

Endpoint 3:

Description: Updating the Business.
URL: http://127.0.0.1:5000/business
Method: PUT
Input(JSON format):

```
{
```

```
  "Business_ID": "TD",
  "Address": "1234 Toronto",
  "Founding_Date": "1855-03-01",
  "Business_Name": "TD Canada Trust"
}
```


Output: "Business updated successfully"

Endpoint 4:

Description: Get the Business by it's ID.
URL: http://127.0.0.1:5000/business/Business_ID
Method: GET
Input: Business_ID
Output(JSON format):


```
{
  "Business": [
    [
      "TD",
      "1234 Toronto",
      "Mon, 01 Jan 1855 00:00:00 GMT",
      "TD Canada Trust"
    ]
  ]
}
```


Endpoint 5:

Description: Deleting the Business by it's ID.
URL: http://127.0.0.1:5000/business/Business_ID
Method: DELETE
Input: Business_ID
Output: "Business deleted successfully"

Endpoint 6:

Description: Creating new Exchange.
URL: http://127.0.0.1:5000/exchange
Method: POST
Input(JSON format):


```
{
  "Name": "New York Stock Exchange",
  "Location": "New York",
  "Number_of_Tickers": 2799
}
```


Output:"Exchange inserted successfully"

Endpoint 7:

Description: Get all Exchanges.
URL: http://127.0.0.1:5000/exchange
Method: GET
Input: None
Output(JSON format):


```
{
  "Exchange": [
    [
      "NASDAQ",
      "New York",
      3300
    ],
    [
      "New York Stock Exchange",
      "New York",
      2799
    ]
  ]
}
```


Endpoint 8:

Description: Updating the Exchange.
URL: http://127.0.0.1:5000/exchange
Method: PUT
Input(JSON format):


```
{
  "Name": "New York Stock Exchange",
  "Location": "New York",
  "Number_of_Tickers": 2800
}
```


Output: "Exchange updated successfully"

Endpoint 9:

Description: Get the Exchange by it's Name.
URL: http://127.0.0.1:5000/exchange/Name
Method: GET
Input: Name
Output(JSON format):


```
{
  "Exchange": [
    [
      "New York Stock Exchange",
      "New York",
      2799
```

```
    ]
  ]
}
```


Endpoint 10:

Description: Deleting the Exchange by it's Name.
URL: http://127.0.0.1:5000/exchange/Name
Method: DELETE
Input: Name
Output: "Exchange deleted successfully"

Endpoint 11:

Description: Creating new Stock.
URL: http://127.0.0.1:5000/stocks
Method: POST
Input(JSON format):


```
{
  "ID": "TSLA.NASDAQ",
  "Company_ID": "TSLA",
  "Prediction_ID": "1002",
  "Predict_Stock_Price": 1140,
  "Strong_Buy": 1,
  "Rating_Buy": 1,
  "Rating_Sell": 0,
  "Strong_Sell": 0,
  "Rating_Hold": 1,
  "Stock_Price": 1137,
  "Sector": "Capital Goods"
}
```


Output:"Stock inserted successfully"

Endpoint 12:

Description: Get all Stocks.
URL: http://127.0.0.1:5000/stocks
Method: GET
Input: None
Output(JSON format):


```
{
  "Stock": [
    [
      "AAPL.NASDAQ",
      "AAPL",
      "1001",
```

```
    160,
    1,
    1,
    0,
    0,
    1,
    151,
    "Tech"
  ],
  [
    "TSLA.NASDAQ",
    "TSLA",
    "1002",
    1140,
    1,
    1,
    0,
    0,
    1,
    1137,
    "Capital Goods"
  ]
 ]
}
```


Endpoint 13:

Description: Updating the Stock.
URL: http://127.0.0.1:5000/stocks
Method: PUT
Input(JSON format):


```
{
 "ID": "TSLA.NASDAQ",
 "Company_ID": "TSLA",
 "Prediction_ID": "1002",
 "Predict_Stock_Price": 1138,
 "Strong_Buy": 1,
 "Rating_Buy": 1,
 "Rating_Sell": 0,
 "Strong_Sell": 0,
 "Rating_Hold": 1,
 "Stock_Price": 1137,
 "Sector": "Capital Goods"
}
```


Output: "Stock updated successfully"

Endpoint 14:
Description: Get the Stock by it's ID.
URL: http://127.0.0.1:5000/stocks/ID
Method: GET
Input: ID
Output(JSON format):

```
{
  "Stock": [
    [
      "TSLA.NASDAQ",
      "TSLA",
      "1002",
      1140,
      1,
      1,
      0,
      0,
      1,
      1137,
      "Capital Goods"
    ]
  ]
}
```


Endpoint 15:
Description: Deleting the Stock by it's ID.
URL: http://127.0.0.1:5000/stocks/ID
Method: DELETE
Input: ID
Output: "Stock deleted successfully"

Endpoint 16:
Description: Creating Belongs To Relationship.
URL: http://127.0.0.1:5000/belongsto
Method: POST
Input(JSON format):

```
{
  "ID": "TSLA.NASDAQ",
  "Name": "NASDAQ"
}
```

Output:"BelongsTo inserted successfully"

Endpoint 17:
Description: Get all BelongsTo.
URL: http://127.0.0.1:5000/belongsto
Method: GET
Input: None
Output(JSON format):

```
{
  "Belongs To": [
    [
      "TSLA.NASDAQ",
      "NASDAQ"
    ]
  ]
}
```


Endpoint 18:
Description: Updating the BelongsTo.
URL: http://127.0.0.1:5000/belongsto
Method: PUT
Input(JSON format):

```
{
  "ID": "TSLA.NASDAQ",
  "Name": "New York Stock Exchange"
}
```

Output: "BelongsTo updated successfully"

Endpoint 19:
Description: Get the BelongsTo by the stock ID.
URL: http://127.0.0.1:5000/belongsto/ID
Method: GET
Input: Stock ID
Output(JSON format):

```
{
  "Belongs To": [
    [
      "TSLA.NASDAQ",
      "NASDAQ"
    ]
  ]
}
```


Endpoint 20:
Description: Deleting the BelongsTo by the stock ID.
URL: http://127.0.0.1:5000/belongsto/ID

Method: DELETE
Input: Stock ID
Output: "BelongsTo deleted successfully"

Endpoint 21:
Description: Creating Offering.
URL: http://127.0.0.1:5000/offering
Method: POST
Input(JSON format):

```
{
  "Offering_ID": "100A",
  "ID": "TSLA.NASDAQ",
  "Quantity_of_stock": 2,
  "Price_offered_at": 1137.01,
  "Status_Complete": "Yes",
  "Status_Incomplete": "N/A"
}
```

Output:"Offering inserted successfully"

Endpoint 22:
Description: Get all Offering.
URL: http://127.0.0.1:5000/offering
Method: GET
Input: None
Output(JSON format):

```
{
  "Offering": [
    [
      "100A",
      "TSLA.NASDAQ",
      2,
      1137.01,
      "Yes",
      "N/A"
    ]
  ]
}
```


Endpoint 23:
Description: Updating the Offering.
URL: http://127.0.0.1:5000/offering
Method: PUT
Input(JSON format):

{
"Offering_ID": "100A",
"ID": "TSLA.NASDAQ",
"Quantity_of_stock": 2,
"Price_offered_at": 1139.01,
"Status_Complete": "Yes",
"Status_Incomplete": "N/A"
}
Output: "Offering updated successfully"

Endpoint 24:
Description: Get the Offering by the Offering ID.
URL: http://127.0.0.1:5000/offering/Offering_ID
Method: GET
Input: Offering ID
Output(JSON format):
{
"Offering": [
[
"100A",
"TSLA.NASDAQ",
2,
1139.01,
"Yes",
"N/A"
]
]
}

Endpoint 25:
Description: Deleting the Offering by the Offering ID.
URL: http://127.0.0.1:5000/offering/Offering_ID
Method: DELETE
Input: Offering ID
Output: "Offering deleted successfully"

Endpoint 26:
Description: Creating an Analyst.
URL: http://127.0.0.1:5000/analyst
Method: POST
Input(JSON format):
{
"Analyst_ID_Number": "AE10",

```
  "ID": "TSLA.NASDAQ",
  "Name": "Peter Parker",
  "Company": "Accenture"
}
```

Output:"Analyst inserted successfully"

Endpoint 27:

Description: Get all Analyst.
URL: http://127.0.0.1:5000/analyst
Method: GET
Input: None
Output(JSON format):


```
{
  "Analyst": [
    [
      "AE10",
      "TSLA.NASDAQ",
      "Peter Parker",
      "Accenture"
    ]
  ]
}
```


Endpoint 28:

Description: Updating the Analyst.
URL: http://127.0.0.1:5000/analyst
Method: PUT
Input(JSON format):


```
{
  "Analyst_ID_Number": "AE10",
  "ID": "TSLA.NASDAQ",
  "Name": "Will Smith",
  "Company": "Accenture"
}
```

Output: "Analyst updated successfully"

Endpoint 29:

Description: Get the Analyst by the Analyst ID Number.
URL: http://127.0.0.1:5000/analyst/Analyst_ID_Number
Method: GET
Input: Analyst ID Number
Output(JSON format):


```
{
  "Analyst": [
```

```
    [
      "AE10",
      "TSLA.NASDAQ",
      "Will Smith",
      "Accenture"
    ]
  ]
}
```


Endpoint 30:

Description: Deleting the Analyst by the Analyst ID Number.
URL: http://127.0.0.1:5000/analyst/Analyst_ID_Number
Method: DELETE
Input: Analyst ID Number
Output: "Analyst deleted successfully"

Endpoint 31:

Description: This call is used to create a new user in the USER table using username as the key
URL: http://127.0.0.1:5000/newuser
Method: POST
Input:(JSON format):


```
{
  "Username": "TestUser",
  "Password": "abcdef",
  "Permissions": "Test"
}
```


Output:: "New User Created"

Endpoint 32:

Description: This API call returns a specific users information (username, password and permissions)
URL: http://127.0.0.1:5000/user/Laura
Method: GET
Input:(JSON format): None
Output: "That User does not exist"
OR
Output(JSON format:


```
{
    "username": [
        [
            "Laura",
            "abcdef",
            "Test"
```

```
      ]
   ]
}
```

Endpoint 33:
Description:This API call is used to update a users password in the USER table
URL: http://127.0.0.1:5000/user/Laura
Method: PUT
Input:(JSON format):

```
 {
   "Password": "laura"
}
```

Output:"Password updated successfully"

Endpoint 34:
Description: This API call us used to delete a user and their information from the USER table
URL: http://127.0.0.1:5000/user/TestUser
Method: DEL
Input:(JSON format):

```
 {
   "Password": "laura"
}
```

Output:"User deleted successfully"
OR
Output: "That User does not exist"

Endpoint 35:
Description: This API call is used to create a New watchlist in the WATCHLIST table
URL: http://127.0.0.1:5000/newWatchlist
Method: POST
Input:(JSON format):

```
 {
   "List_Number": "4"
}
```

Output:"New Watchlist Created"

Endpoint 36:
Description: This API call adds a new stock to the CONTAINS table where the watchlist ID used as a key. It will return either successfully added or that the watchlist is already in the list
URL: http://127.0.0.1:5000/watchlist/4
Method: POST
Input:(JSON format):

{
"Stock_ID": "AAPL.NASDAQ"
}

Output(JSON format):
{
"AAPL.NASDAQ": "is already in Watchlist"
}
OR
Output(JSON format)
{
"AAPL.NASDAQ": "has been added to WatchList"
}

Endpoint 37:
Description: This API call deletes a stock from the CONTAINS table where the watchlist ID used as a key to reference the WATCHLIST table
URL: http://127.0.0.1:5000/watchlist/4
Method: DEL
Input:(JSON format):
{
"Stock_ID": "AAPL.NASDAQ"
}

Output(JSON format):
{
"AAPL.NASDAQ": "has been deleted successfully from your Watchlist"
}

Endpoint 38:
Description: This API returns stocks from the CONTAINS table where the watchlist ID used as a key to reference the WATCHLIST table
URL: http://127.0.0.1:5000/watchlist/1
Method: GET
Input:(JSON format): None

Output(JSON format):
{
"watchlist_ID": [
[
1,
"BBBL.NASDAQ"
],

```
      [
         1,
         "AAPL.NASDAQ"
      ]
   ]
}
```


Endpoint 39:

Description: This API call returns all users information that is entered in the USERS table or specifies that there are no users in the table

URL: http://127.0.0.1:5000/admin/usersall

Method: GET

Input:(JSON format): None

Output(JSON format):

"No Users In Data Base"

OR

Output(JSON format):


```
{
   "username": [
      [
         "abhay",
         "abcdef",
         "Test"
      ],
      [
         "Laura",
         "abcdef",
         "Test"
      ],
      [
         "TestUser",
         "abcdef",
         "Test"
      ]
   ]
}
```


Endpoint 40:

Description: This API call is used to update a specific users permission in the USER table

URL:http://127.0.0.1:5000/admin

Method: PUT

Input:(JSON format):


```
{
   "Permissions": "TestChange",
   "Username": "abhay"
```

}

Output(JSON format):
"Permissions updated successfully"

Endpoint 41:
Description: This API call is used to delete a specific user from the USER database
URL:http://127.0.0.1:5000/admin
Method: DEL
Input:(JSON format):
{

"Username": "abhay"

}

Output(JSON format):
"User deleted successfully"
OR
Output(JSON format):
"That User does not exist"

Endpoint 42:
Description: This API call is used to create a private user in the PRIVATE table
URL:http://127.0.0.1:5000/private
Method: POST
Input:(JSON format):
{
"Username": "Laura",
"List_Number": "",
"Role_Type" : "Private"
}

Output(JSON format):
{
"Laura": "has been added to Private"
}

Endpoint 43:
Description: This API call is used to delete a private user from the PRIVATE table
URL:http://127.0.0.1:5000/private
Method: DEL
Input:(JSON format):

{
"Username": "Laura"

}

Output(JSON format):
"User deleted successfully"
OR
Output(JSON format):
"That User does not exist"

Endpoint 44:
Description: This API call is used to return the details for a specific user (username, watchlist, and role) from the PRIVATE table
URL:http://127.0.0.1:5000/private
Method: GET
Input:(JSON format):
{
"Username": "Laura"

}

Output(JSON format):
{
"UserDetails": [
[
"Laura",
2,
"Private"
]
]
}

Endpoint 45:
Description: This API call is used to update the watchlist number that the user is watching
URL:http://127.0.0.1:5000/private
Method: PUT
Input:(JSON format):
{
"Username": "Laura",
"List_Number": "5"

}

Output(JSON format):
"WatchList updated successfully"

Endpoint 46:
Description:This API call is used to create a Professional Account in the PROFESSIONAL table
URL:http://127.0.0.1:5000/professional
Method: POST
Input:(JSON format):
{
"Username": "Laura",
"List_Number": "1",
"Role_Type" : "professional"
}

Output(JSON format):
{
"Laura": "has been added to Professional"
}

Endpoint 47:
Description: This API call is used to delete a Professional user from the PROFESSIONAL table
URL:http://127.0.0.1:5000/professional
Method: DEL
Input:(JSON format):
{
"Username": "Laura"

}

Output(JSON format):
"User deleted successfully"
OR
Output(JSON format):
"That User does not exist"

Endpoint 48:
Description: This API call is used to return the details for a specific user (username, watchlist, and role) from the PROFESSIONAL table
URL:http://127.0.0.1:5000/professional
Method: GET
Input:(JSON format):
{

"Username": "Laura"

}

Output(JSON format):

{
"UserDetails": [
[
"Laura",
1,
"professional"
]
]
}

Endpoint 49:

Description: This API call is used to update the watchlist number that the professional user is watching
URL: http://127.0.0.1:5000/professional
Method: PUT
Input:(JSON format):

{
"Username": "Laura",
"List_Number": "4"

}

Output(JSON format):
"WatchList updated successfully"

Endpoint 50:

Description: Creating new Stockevent.
URL: http://127.0.0.1:5000/stockevent
Method: POST
Input(JSON format):

{
"Event_ID": "1002",
"Stock_ID": "AAPL.NASDAQ",
"Time": "12:00:00",
"Date": "2022-07-24",
"P_ID": "1001",
"Bearish_sentiment": "0",
"Neutral_sentiment": "0",
"Bullish_sentiment": "1",

```
  "Price_Change": 4,
  "Predict_Stock_Events": "Null"
}
```


Output:"Stockevent INSERTED successfully"

Endpoint 51:

Description: Get all Stockevent.
URL: http://127.0.0.1:5000/stockevent
Method: GET
Input: None
Output(JSON format):


```
{
 "Stockevent": [
   [
          "1002",
          "AAPL.NASDAQ",
           "12:00:00",
          "2022-07-24",
          "1001",
          "0",
          "0",
          "1",
          4,
           "Null"
   ]
 ]
}
```


Endpoint 52:

Description: Updating the Stockevent.
URL: http://127.0.0.1:5000/stockevent
Method: PUT
Input(JSON format):


```
{
  "Event_ID": "1002",
  "Stock_ID": "AAPL.NASDAQ",
  "Time": "12:00:00",
  "Date": "2022-07-24",
  "P_ID": "1001",
  "Bearish_sentiment": "10",
  "Neutral_sentiment": "10",
  "Bullish_sentiment": "1000",
```

"Price_Change": 6,
"Predict_Stock_Events": "Null"
}

Output: "Stockevent updated successfully"

Endpoint 53:

Description: Creating new PR.
URL: http://127.0.0.1:5000/pr
Method: POST
Input(JSON format):

{
"Event_ID": "1002",
"P_ID": "1001",
"Headline": "New Highs for Apple",
"Predict_Stock_Events": "Null"
}

Output:"PR INSERTED successfully"

Endpoint 54:

Description: Get all PR.
URL: http://127.0.0.1:5000/PR
Method: GET
Input: None
Output(JSON format):

{
"PR": [
[
"1002",
"1001",
"New Highs for Apple",
"Null"
]
]
}

Endpoint 55:

Description: Updating the PR.
URL: http://127.0.0.1:5000/pr
Method: PUT
Input(JSON format):

{
"Event_ID": "1002",

```
  "P_ID": "1001",
  "Headline": "New Highs for Apple 3T MK",
  "Predict_Stock_Events": "Null"
}
```


Output: "PR updated successfully"

Endpoint 56:

Description: Creating new Prediction.
URL: http://127.0.0.1:5000/prediction
Method: POST
Input(JSON format):


```
{
  "P_ID": "1001"
}
```


Output:"Prediction INSERTED successfully"

Endpoint 57:

Description: Get all Prediction.
URL: http://127.0.0.1:5000/prediction
Method: GET
Input: None
Output(JSON format):


```
{
 "Prediction": [
   [
         "1001"
   ]
 ]
}
```


Endpoint 58:

Description: Updating the Prediction.
URL: http://127.0.0.1:5000/prediction
Method: PUT
Input(JSON format):


```
{
  "P_ID": "1011",
}
```


Output: "Prediction updated successfully"

Endpoint 59:

Description: Creating new Secfiling.

URL: http://127.0.0.1:5000/secfiling
Method: POST
Input(JSON format):

{
"Event_ID": "1002",
"P_ID": "1001",
"Type_of_Filing": "10-K",
"Predict_Stock_Events": "Null"
}

Output:"Secfiling INSERTED successfully"

Endpoint 60:
Description: Get all Secfiling.
URL: http://127.0.0.1:5000/secfiling
Method: GET
Input: None
Output(JSON format):

{
"Secfiling": [
[
"1002",
"1001",
"10-K",
"Null"
]
]
}

Endpoint 61:
Description: Updating the Secfiling.
URL: http://127.0.0.1:5000/secfiling
Method: PUT
Input(JSON format):

{
"Event_ID": "1002",
"P_ID": "1001",
"Type_of_Filing": "8-K",
"Predict_Stock_Events": "Null"
}

Output: "Secfiling updated successfully"

Endpoint 62:

Description: Creating new 52 Week.
URL: http://127.0.0.1:5000/week52
Method: POST
Input(JSON format):


```
{
  "Event_ID": "1002",
  "P_ID": "1001",
  "Value_1": "159",
  "Type_High": "1",
  "Type_Low": "0",
  "Predict_52Week": "Null"
}
```


Output:"52 Week INSERTED successfully"

Endpoint 63:

Description: Get all 52 Week.
URL: http://127.0.0.1:5000/week52
Method: GET
Input: None
Output(JSON format):


```
{
 "Week52": [
   [
          "1002",
          "1001",
          "0",
          "0",
          "1",
         "Null"
   ]
 ]
}
```


Endpoint 64:

Description: Updating the 52 Week.
URL: http://127.0.0.1:5000/week52
Method: PUT
Input(JSON format):


```
{
  "Event_ID": "1002",
```

"P_ID": "1001",
"Value_1": "160",
"Type_High": "1",
"Type_Low": "0",
"Predict_52Week": "Null"
}

Output: "Week52 updated successfully"