Placecon Options Chain

TEAM

Niraj Patil - 16010320052
Nitai Kodkani - 16010120080
Harsh Rathi - 16010320058
Saikrishna Karra - 16010120075
Ved Shrirao - 16010320014

Problem Statement

BUILD AN OPTIONS CHAIN TOOL

INTRODUCTION

- DEVELOP A REAL-TIME OPTIONS CHAIN TOOL SIMILAR TO NSE INDIA'S PLATFORM.
- PROCESS MARKET DATA AND DYNAMICALLY UPDATE THE TOOL WITHOUT REQUIRING A PAGE RELOAD.
- CALCULATE IMPLIED VOLATILITY (IV) USING THE BLACK SCHOLES FORMULA, CONSIDERING UNDERLYING PRICE, STRIKE PRICE, TIME TO MATURITY (TTM), AND A 5% RISK-FREE INTEREST RATE.
- DISPLAY AN OPTIONS CHAIN WITH A SELECTION OF UNDERLYING ASSETS AND DIFFERENT EXPIRY DATES FOR USER ANALYSIS.
- ACCURATELY CALCULATE TTM ASSUMING EXPIRY TIME AT 15:30 IST ON THE EXPIRY DAY, WITH IV BECOMING 0 IMMEDIATELY AFTER EXPIRATION.
- UTILIZE A PUBLISHED MARKET DATA STREAM TO OBTAIN THE UNDERLYING PRICE FOR CALCULATIONS AND DISPLAY.
- BY FULFILLING THESE REQUIREMENTS, THE OPTIONS CHAIN TOOL WILL PROVIDE USERS WITH REAL-TIME MARKET DATA, IMPLIED VOLATILITY CALCULATIONS, AND A COMPREHENSIVE VIEW OF AVAILABLE OPTIONS FOR ANALYSIS AND DECISION-MAKING.

OPTION CHAIN

- OPTIONS ARE FINANCIAL DERIVATIVES THAT GIVE THE HOLDER THE RIGHT, BUT NOT THE OBLIGATION, TO BUY OR SELL AN UNDERLYING ASSET AT A PREDETERMINED PRICE (KNOWN AS THE STRIKE PRICE) ON OR BEFORE A SPECIFIC DATE (KNOWN AS THE EXPIRATION DATE).

- THESE UNDERLYING ASSETS CAN INCLUDE STOCKS, COMMODITIES, INDICES, OR CURRENCIES.

- OPTIONS PROVIDE TRADERS AND INVESTORS WITH A FLEXIBLE AND VERSATILE TOOL TO MANAGE RISK, SPECULATE ON PRICE MOVEMENTS, OR ENHANCE PORTFOLIO STRATEGIES.

- AN OPTION CHAIN IS A COMPREHENSIVE TABLE OR LIST THAT DISPLAYS ALL AVAILABLE OPTIONS CONTRACTS FOR A SPECIFIC UNDERLYING ASSET.

- IT PROVIDES TRADERS AND INVESTORS WITH A CLEAR VIEW OF THE VARIOUS STRIKE PRICES AND EXPIRATION DATES FOR BOTH CALL AND PUT OPTIONS RELATED TO THE CHOSEN UNDERLYING ASSET.

CALL & PUT

CALL

- A CALL OPTION IS A TYPE OF FINANCIAL CONTRACT THAT GIVES THE HOLDER (BUYER) THE RIGHT, BUT NOT THE OBLIGATION, TO BUY AN UNDERLYING ASSET AT A PREDETERMINED PRICE (STRIKE PRICE) ON OR BEFORE A SPECIFIC DATE (EXPIRATION DATE).
- THE BUYER OF A CALL OPTION BELIEVES THAT THE PRICE OF THE UNDERLYING ASSET WILL INCREASE IN THE FUTURE. BY PURCHASING A CALL OPTION, THE BUYER HAS THE OPPORTUNITY TO PROFIT FROM THE POTENTIAL PRICE APPRECIATION OF THE ASSET.

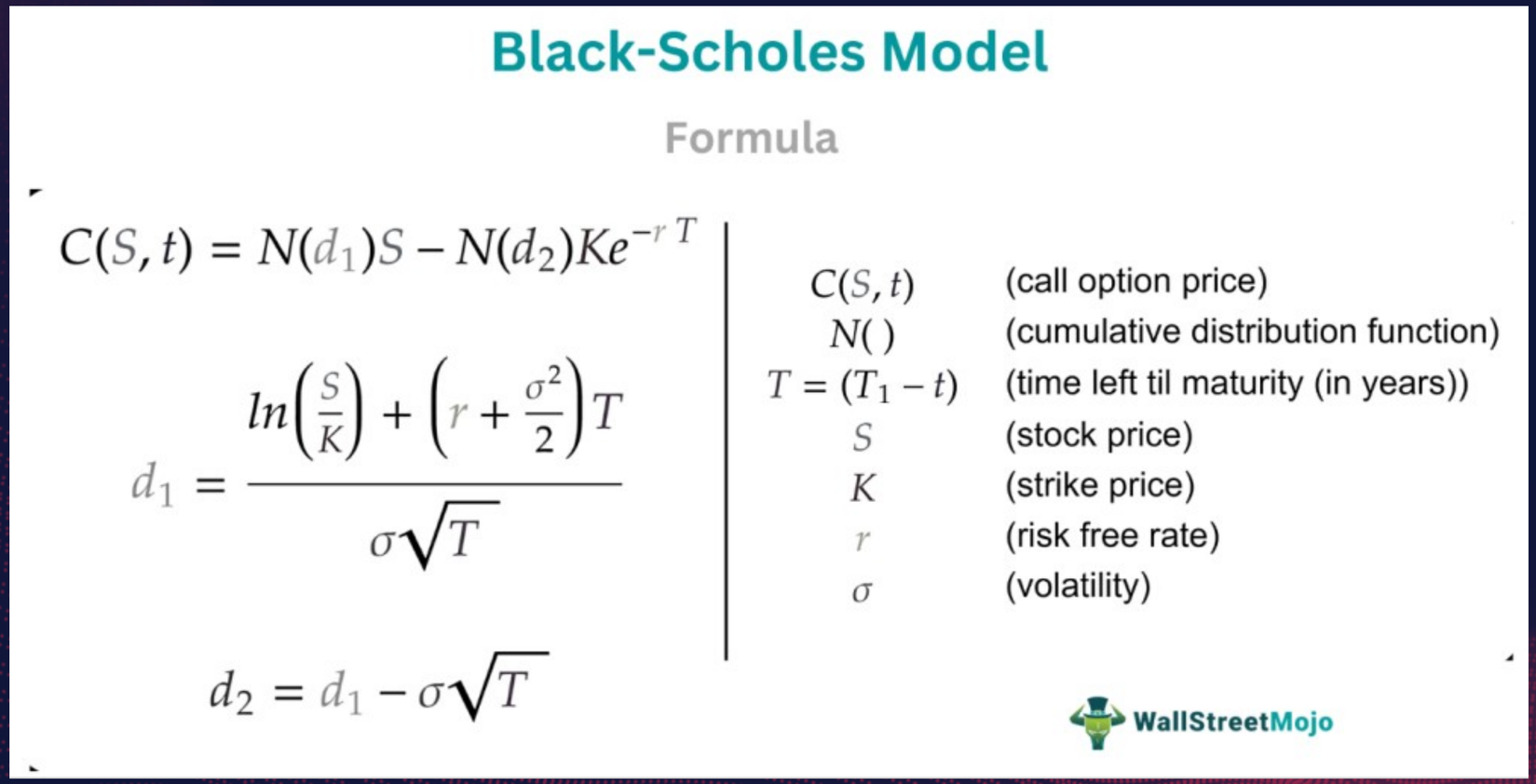
PUT

- ON THE OTHER HAND, A PUT OPTION IS A FINANCIAL CONTRACT THAT PROVIDES THE HOLDER (BUYER) WITH THE RIGHT, BUT NOT THE OBLIGATION, TO SELL AN UNDERLYING ASSET AT A PREDETERMINED PRICE (STRIKE PRICE) ON OR BEFORE A SPECIFIC DATE (EXPIRATION DATE).
- THE BUYER OF A PUT OPTION EXPECTS THE PRICE OF THE UNDERLYING ASSET TO DECREASE. BY PURCHASING A PUT OPTION, THE BUYER HAS THE OPPORTUNITY TO PROFIT FROM A POTENTIAL DECLINE IN THE ASSET'S PRICE.

IMPLIED VOLATILITY (IV)

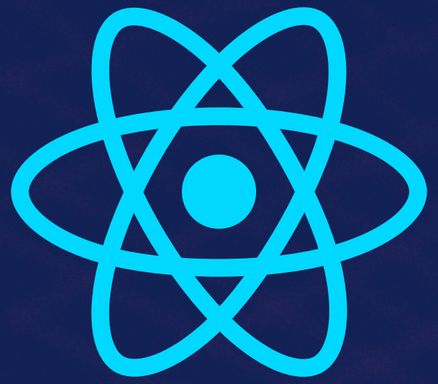
- IMPLIED VOLATILITY (IV) IS A MEASURE OF THE MARKET'S EXPECTATIONS FOR FUTURE PRICE VOLATILITY OF AN UNDERLYING ASSET.
- IT IS A CRITICAL PARAMETER USED IN OPTIONS PRICING MODELS, SUCH AS THE BLACK-SCHOLES MODEL, AND PLAYS A SIGNIFICANT ROLE IN DETERMINING OPTION PRICES.
- OPTIONS PROVIDE TRADERS AND INVESTORS WITH A FLEXIBLE AND VERSATILE TOOL TO MANAGE RISK, SPECULATE ON PRICE MOVEMENTS, OR ENHANCE PORTFOLIO STRATEGIES.
- IMPLIED VOLATILITY REFLECTS THE COLLECTIVE SENTIMENT AND UNCERTAINTY OF MARKET PARTICIPANTS REGARDING THE FUTURE MOVEMENT OF THE UNDERLYING ASSET'S PRICE.
- IT REPRESENTS THE MARKET'S ESTIMATE OF THE POTENTIAL MAGNITUDE OF PRICE SWINGS OR FLUCTUATIONS OVER THE OPTION'S REMAINING LIFESPAN.

OUR APPROACH OF IV



THE CALCULATE_IMPLIED_VOLATILITY FUNCTION USES THE BLACK-SCHOLES-MERTON MODEL TO ESTIMATE THE IMPLIED VOLATILITY OF AN OPTION. IT ITERATIVELY ADJUSTS THE VOLATILITY PARAMETER (SIGMA) UNTIL THE CALCULATED OPTION PRICE MATCHES THE OBSERVED MARKET PRICE. THE FUNCTION CALCULATES D1 AND D2 VALUES, APPLIES THE CUMULATIVE DISTRIBUTION FUNCTION (CDF) TO THESE VALUES, AND COMPUTES THE OPTION PRICE BASED ON THE OPTION TYPE. THE IMPLIED VOLATILITY REPRESENTS THE MARKET'S EXPECTATION OF FUTURE VOLATILITY IN THE UNDERLYING ASSET.

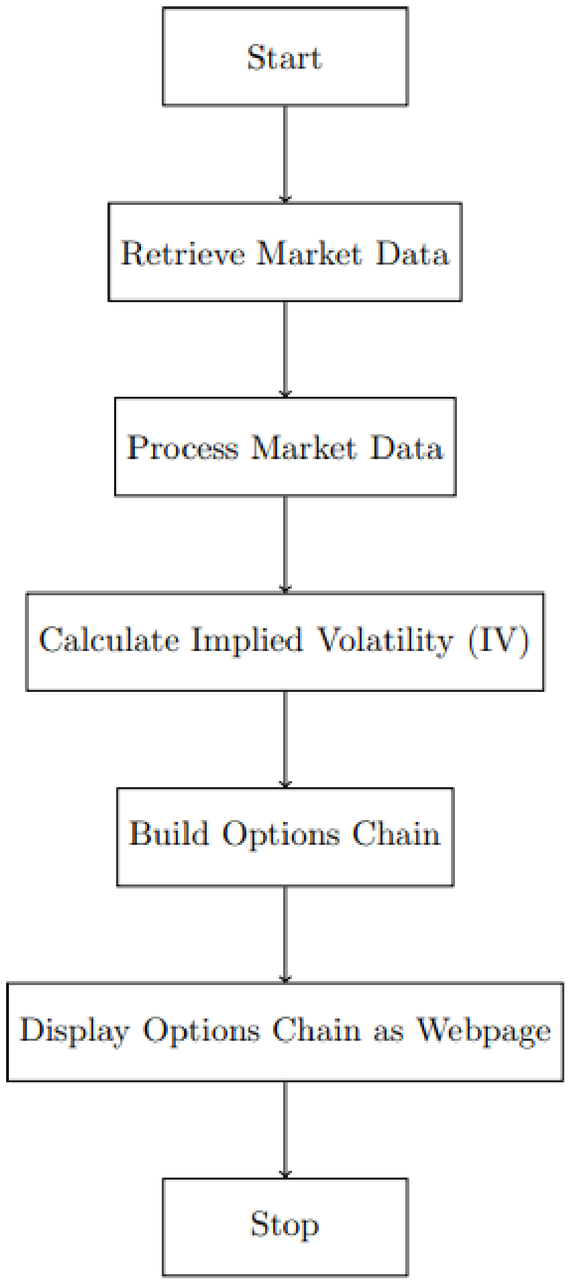
TECH STACK



React

python™

node.js



PROCEDURE

WE START BY IMPORTING THE REQUIRED LIBRARIES AND MODULES.

WE USE THE SUBPROCESS.POPEN FUNCTION TO RUN A JAR FILE AND CAPTURE ITS OUTPUT. THE JAR FILE IS EXECUTED WITH SPECIFIC ARGUMENTS.

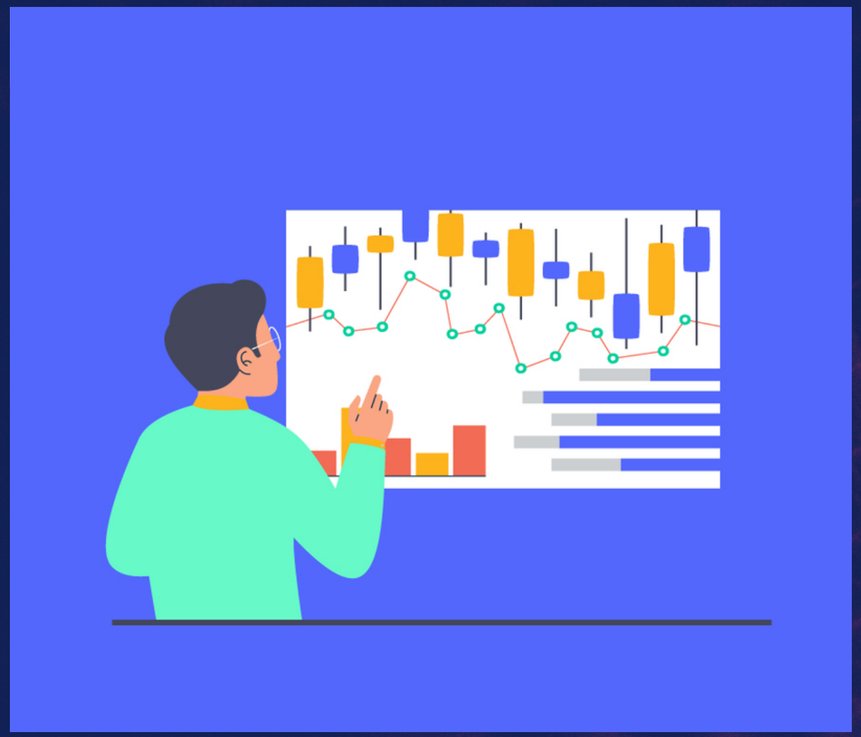
WE INITIALIZE A LIST TO STORE THE DATA AND A VARIABLE TO COUNT THE NUMBER OF ROWS.

WE ENTER A LOOP TO READ AND PROCESS THE CONTINUOUS STREAM OF DATA FROM THE JAR FILE'S OUTPUT.

WITHIN THE LOOP, WE EXTRACT THE RELEVANT DATA FROM THE OUTPUT USING REGULAR EXPRESSIONS.

IF DATA IS FOUND, WE CONVERT IT INTO A DICTIONARY AND APPEND IT TO THE DATA LIST. WE ALSO INCREMENT THE ROW COUNT.

PROCEDURE



EVERY 100 ROWS, WE CREATE A PANDAS DATAFRAME FROM THE COLLECTED DATA.

WE PERFORM DATA PREPROCESSING ON THE DATAFRAME, INCLUDING SEPARATING PARTS OF A COLUMN, CALCULATING TIME TO MATURITY, AND CONVERTING COLUMNS TO NUMERIC TYPES.

WE DEFINE A FUNCTION CALCULATE_IMPLIED_VOLATILITY TO CALCULATE THE IMPLIED VOLATILITY USING THE BLACK-SCHOLES FORMULA.

WE APPEND AN EMPTY COLUMN 'IMPLIEDVOLATILITY' TO THE DATAFRAME.

WE ITERATE OVER EACH ROW IN THE DATAFRAME AND CALCULATE THE IMPLIED VOLATILITY USING AN ITERATIVE METHOD UNTIL CONVERGENCE.

THE CALCULATED IMPLIED VOLATILITY IS ASSIGNED TO THE CORRESPONDING ROW IN THE 'IMPLIEDVOLATILITY' COLUMN.

FINALLY, WE SAVE THE DATAFRAME AS A JSON FILE AND PRINT A SUCCESS MESSAGE.

THE SUBPROCESS IS TERMINATED.

final100.py > ...

```python
def calculate_time_to_maturity(expiry_date):
    if pd.isna(expiry_date) or expiry_date.strip() == '':
        return None

    expiry_time = datetime.strptime(expiry_date, '%d%b')
    current_time = datetime.now(pytz.timezone('Asia/Kolkata'))
    expiry_datetime = datetime(current_time.year, expiry_time.month, expiry_time.day, 15, 30, tzinfo=
    if current_time > expiry_datetime:
        expiry_datetime += timedelta(days=365)
    time_to_maturity = expiry_datetime - current_time
    return round(time_to_maturity.total_seconds() / 86400)

df['TTM'] = df['Expiry Date'].apply(calculate_time_to_maturity)

df['TTM'] = df['TTM'].fillna(0)

def calculate_implied_volatility(S, K, r, T, option_price, option_type, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    if option_type == 'call':
        N_d1 = norm.cdf(d1)
        N_d2 = norm.cdf(d2)
        option_price_calc = S * N_d1 - K * np.exp(-r * T) * N_d2
    elif option_type == 'put':
        N_minus_d1 = norm.cdf(-d1)
        N_minus_d2 = norm.cdf(-d2)
        option_price_calc = K * np.exp(-r * T) * N_minus_d2 - S * N_minus_d1

    return option_price_calc - option_price
```

Live Share Ln 59, Col 67 Spaces: 4 UTF-8

output.json > ...

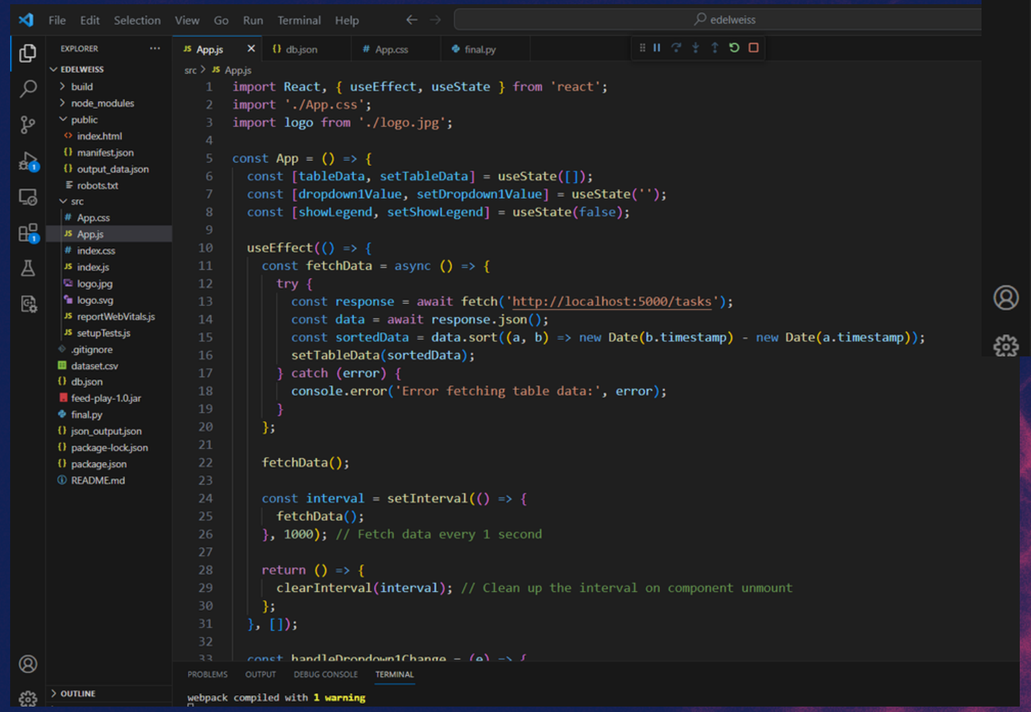

```json
{
    "tasks": [
        {
            "symbol": "'ALLBANKS27JUL2345100PE'",
            "Stock": "ALLBANKS",
            "Expiry Date": "27JUL",
            "Strike price": "2345100",
            "Put/Call": "'PE'",
            "LTP": 132130,
            "LTQ": "0",
            "totalTradedVolume": "0",
            "bestBid": 116270,
            "bestAsk": "122245",
            "bestBidQty": "50",
            "bestAskQty": "50",
            "openInterest": 0,
            "timestamp": "Tue Jul 04 23:03:14 IST 2023",
            "sequence": "1301",
```

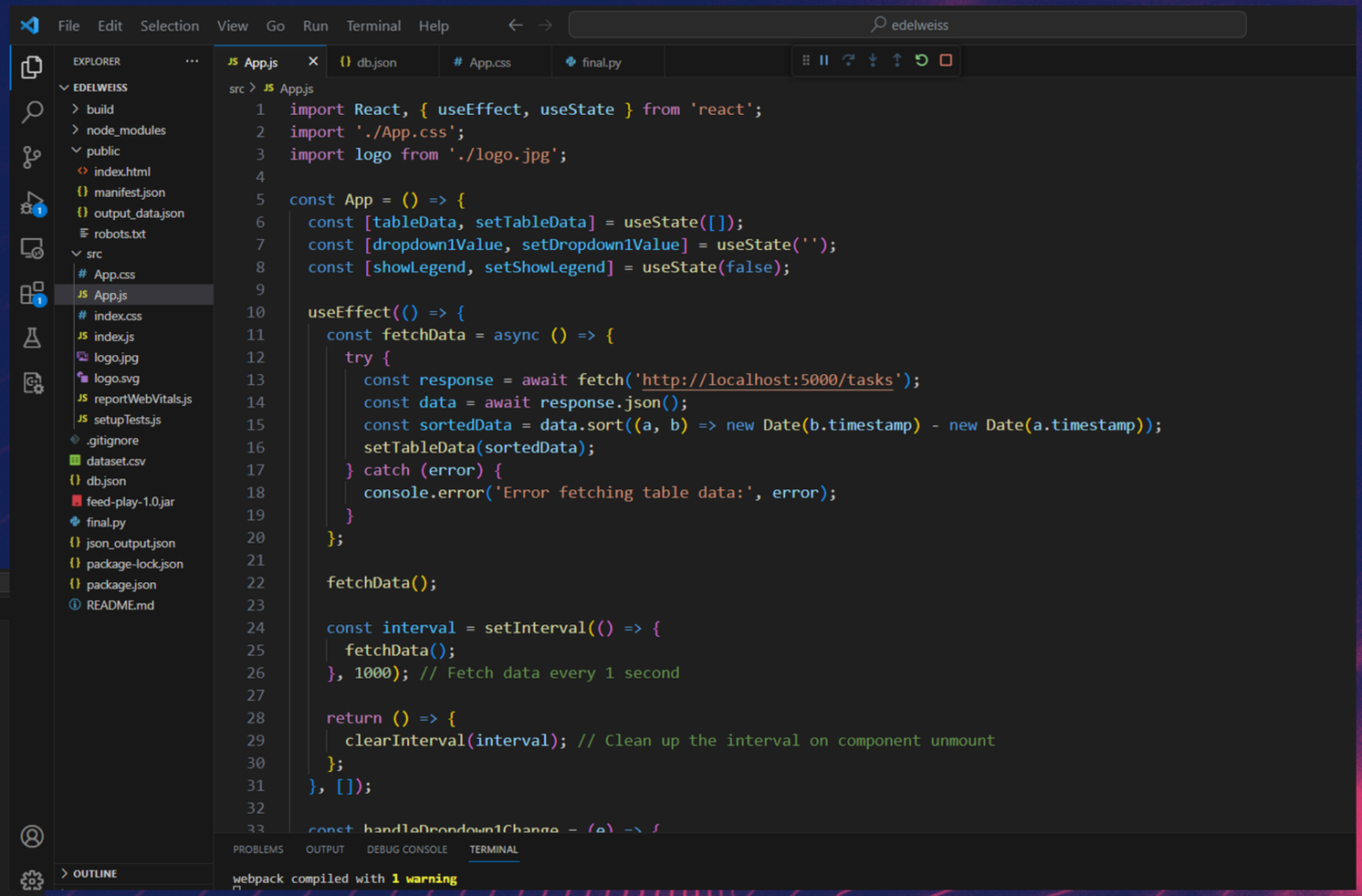

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: python

```
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
JSON file created successfully!
```

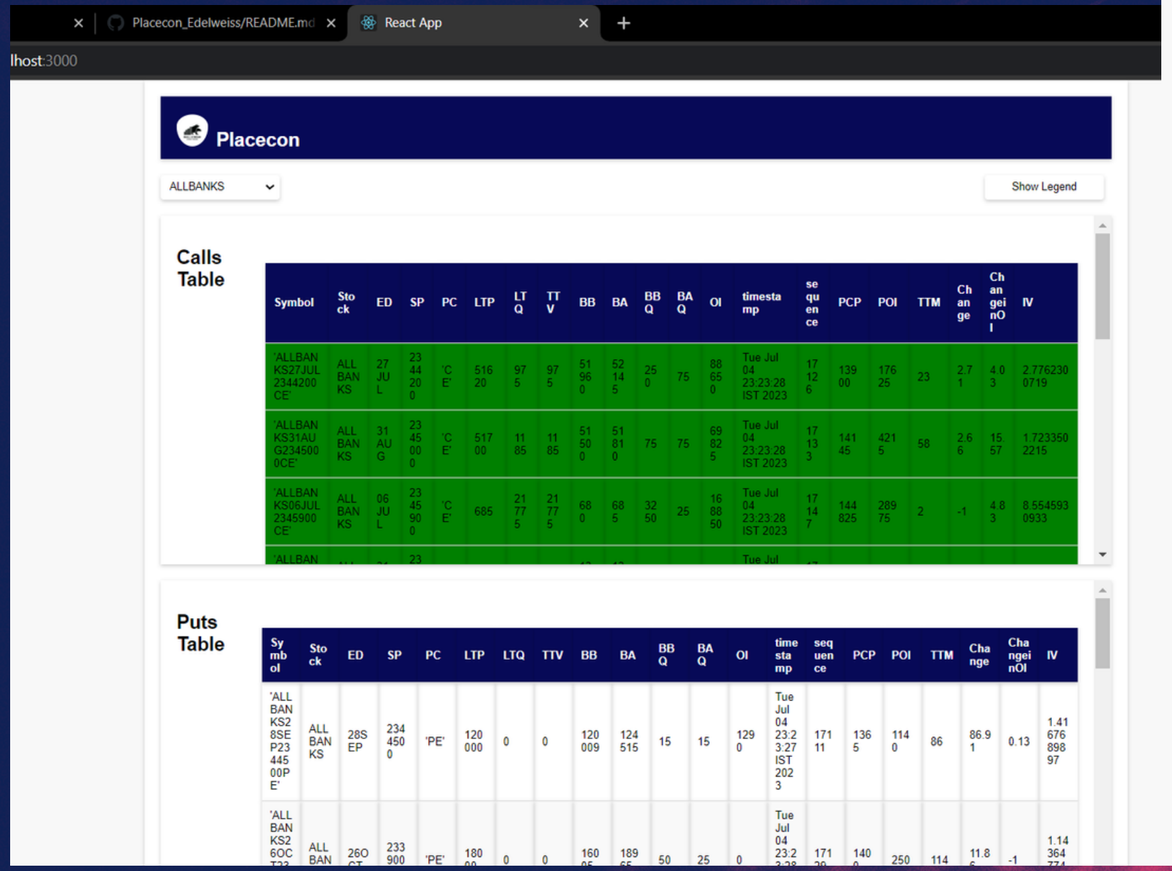
Live Share Ln 1, Col 1 Spaces: 4 UTF-8 CRLF JSON Go Live

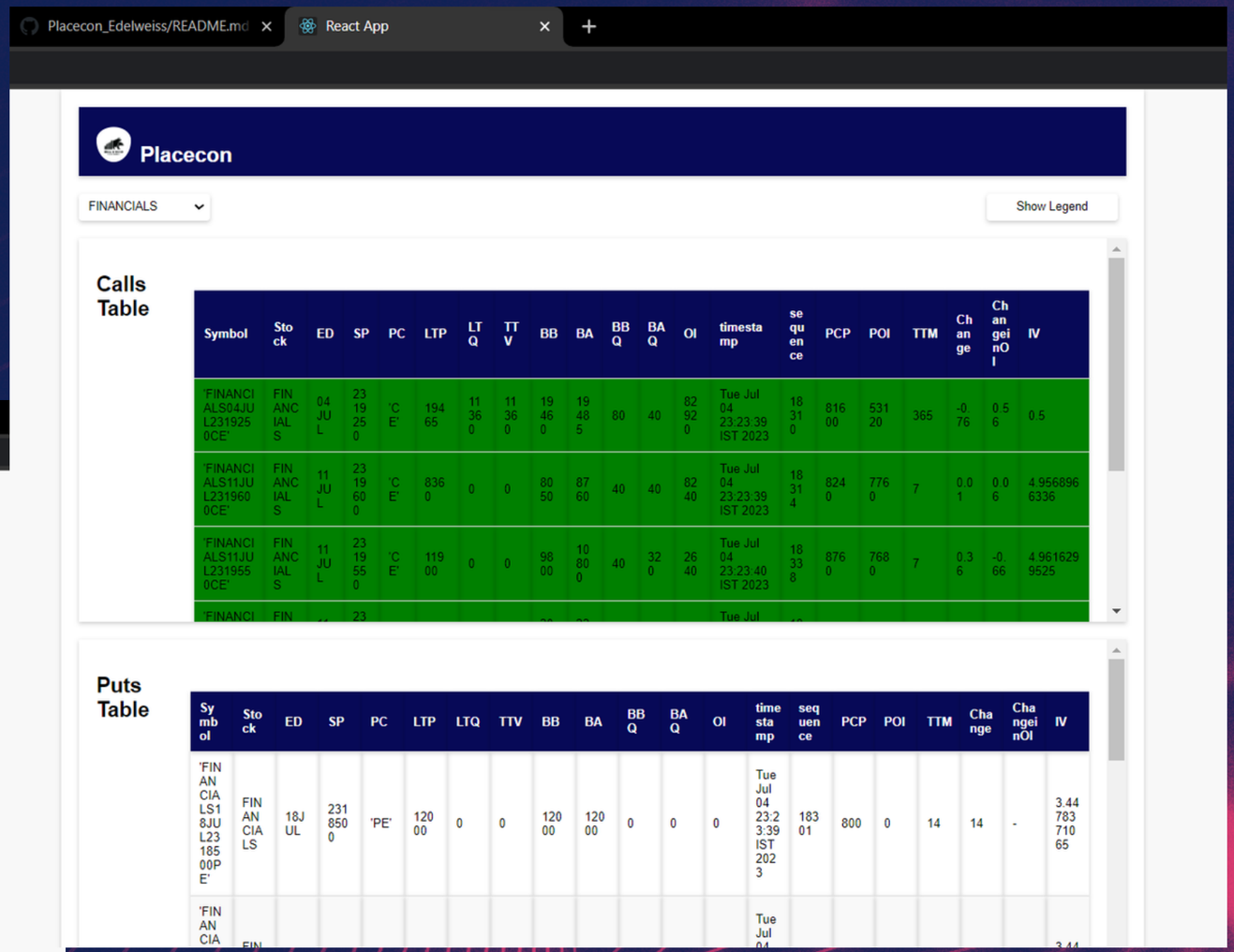
PROCEDURE


File Edit Selection View Go Run Terminal Help
edelweiss
EXPLORER
EDELWEISS
build
node_modules
public
index.html
manifest.json
output_data.json
robots.txt
src
App.css
App.js
index.css
index.js
logo.jpg
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
dataset.csv
db.json
feed-play-1.0.jar
final.py
json_output.json
package-lock.json
package.json
README.md
App.js
db.json
App.css
final.py
src > App.js
import React, { useEffect, useState } from 'react';
import './App.css';
import logo from './logo.jpg';
const App = () => {
const [tableData, setTableData] = useState([]);
const [dropdown1Value, setDropdown1Value] = useState('');
const [showLegend, setShowLegend] = useState(false);
useEffect(() => {
const fetchData = async () => {
try {
const response = await fetch('http://localhost:5000/tasks');
const data = await response.json();
const sortedData = data.sort((a, b) => new Date(b.timestamp) - new Date(a.timestamp));
setTableData(sortedData);
} catch (error) {
console.error('Error fetching table data:', error);
}
};
fetchData();
const interval = setInterval(() => {
fetchData();
}, 1000); // Fetch data every 1 second
return () => {
clearInterval(interval); // Clean up the interval on component unmount
};
}, []);
OUTLINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
webpack compiled with 1 warning


File Edit Selection View Go Run Terminal Help
edelweiss
EXPLORER
EDELWEISS
build
node_modules
public
index.html
manifest.json
output_data.json
robots.txt
src
App.css
App.js
index.css
index.js
logo.jpg
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
dataset.csv
db.json
feed-play-1.0.jar
final.py
json_output.json
package-lock.json
package.json
README.md
App.js
db.json
App.css
final.py
src > App.js
import React, { useEffect, useState } from 'react';
import './App.css';
import logo from './logo.jpg';
const App = () => {
const [tableData, setTableData] = useState([]);
const [dropdown1Value, setDropdown1Value] = useState('');
const [showLegend, setShowLegend] = useState(false);
useEffect(() => {
const fetchData = async () => {
try {
const response = await fetch('http://localhost:5000/tasks');
const data = await response.json();
const sortedData = data.sort((a, b) => new Date(b.timestamp) - new Date(a.timestamp));
setTableData(sortedData);
} catch (error) {
console.error('Error fetching table data:', error);
}
};
fetchData();
const interval = setInterval(() => {
fetchData();
}, 1000); // Fetch data every 1 second
return () => {
clearInterval(interval); // Clean up the interval on component unmount
};
}, []);
OUTLINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
webpack compiled with 1 warning

RESULT

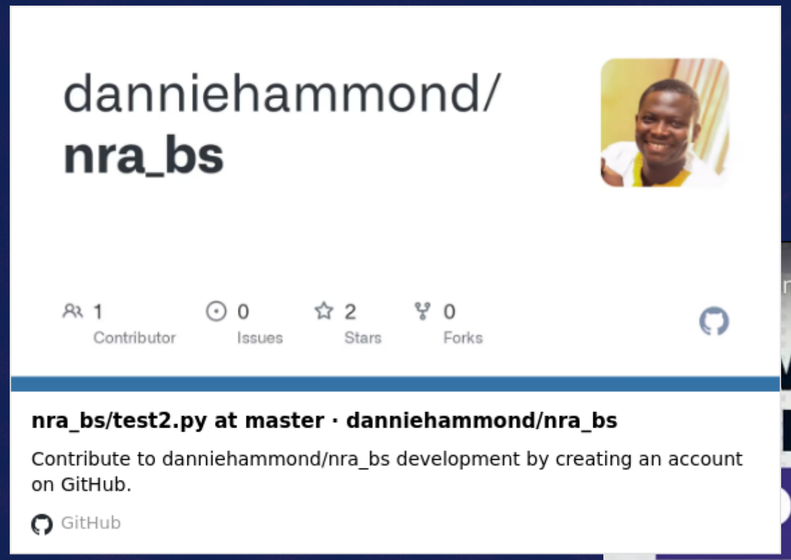

Placecon_Edelweiss/README.md
React App
lhost:3000
Placecon
ALLBANKS
Show Legend
Calls Table
Puts Table


Placecon_Edelweiss/README.md
React App
Placecon
FINANCIALS
Show Legend
Calls Table
Puts Table

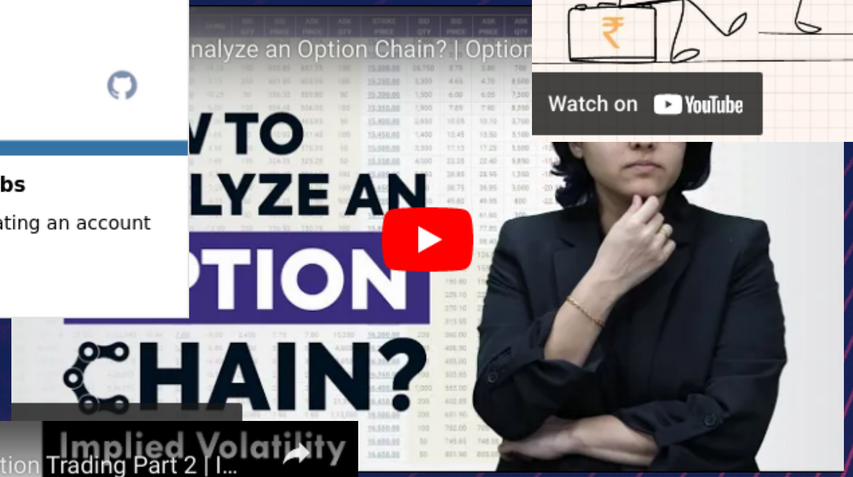
CONCLUSION

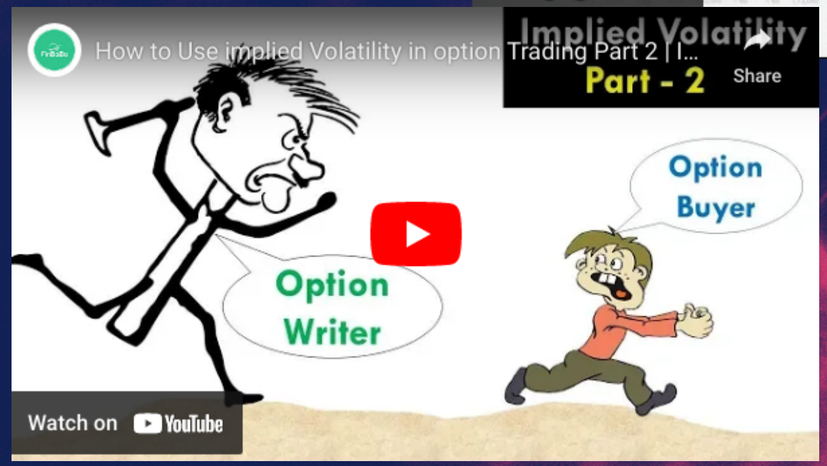
WE HAVE SUCCESSFULLY BUILT A DYNAMIC OPTION CHAIN TOOL THAT PROCESSES MARKET DATA IN REAL TIME AND HAVE BEEN ABLE TO DISPLAY THE TIME TO MATURITY(TTM) AND THE IMPLIED VOLATILITY(IV) ACCURATELY.

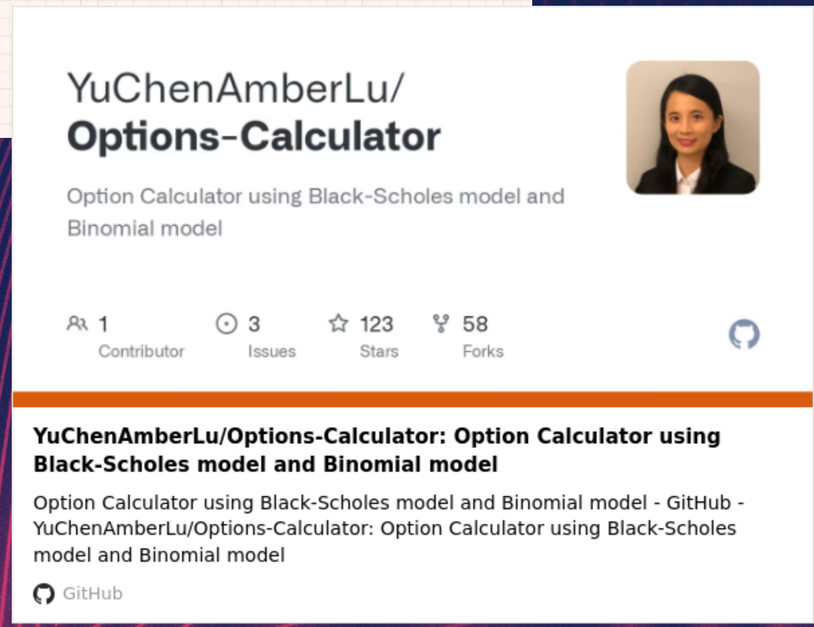
REFERENCES


danniehammond/
nra_bs
1 Contributor
0 Issues
2 Stars
0 Forks
nra_bs/test2.py at master · danniehammond/nra_bs
Contribute to danniehammond/nra_bs development by creating an account on GitHub.
GitHub


Introduction to Options
ZERODHA
VARSITY
Video 1
Introduction to Options
Watch on YouTube


nalyze an Option Chain? | Optio
V TO
LYZE AN
TION
CHAIN?


How to Use implied Volatility in option Trading Part 2 | I...
Implied Volatility
Part - 2
Share
Option Writer
Option Buyer
Watch on YouTube


YuChenAmberLu/
Options-Calculator
Option Calculator using Black-Scholes model and Binomial model
1 Contributor
3 Issues
123 Stars
58 Forks
YuChenAmberLu/Options-Calculator: Option Calculator using Black-Scholes model and Binomial model
Option Calculator using Black-Scholes model and Binomial model - GitHub - YuChenAmberLu/Options-Calculator: Option Calculator using Black-Scholes model and Binomial model
GitHub

THANK YOU!