In [26]:
```
#Group Name - TeenPatti
#Group Members Name and enrollment Number

# 1. Amit Kumar Sharma - M2023ANLT002
# 2. Prashik Bansod - M2023ANLT016
# 3. Saurav Ranjan - M2023ANLT025
# 4. Shahith Ahamed - M2023ANLT026
# 5. Utkarsh Soni - M2023ANLT031
# 6. Vaibhav Ahalawat - M2023ANLT034
# 7. Vivek Kumar - M2023ANLT038
```

In [ ]:
```
```

In [7]:
```
import pandas as pd
import matplotlib.pyplot as plt
import warnings
import yfinance as yf


# Ignore all warnings
warnings.simplefilter("ignore")
```

In [8]:
```
# Define the stock symbol and time period
stock_symbol = "AAPL"
start_date = "2018-01-01"
end_date = "2022-12-31"

# Download daily stock data
data = yf.download(stock_symbol, start=start_date, end=end_date)

print(data.columns)
#stock_data = data[data['Date','Close']]

#print(stock_data)
# Print the downloaded data
print(data.head())
```

```
[**********100%%**********]  1 of 1 completed
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
                 Open       High        Low      Close  Adj Close     Volume
Date
2018-01-02  42.540001  43.075001  42.314999  43.064999  40.722874  102223600
2018-01-03  43.132500  43.637501  42.990002  43.057499  40.715775  118071600
2018-01-04  43.134998  43.367500  43.020000  43.257500  40.904911   89738400
2018-01-05  43.360001  43.842499  43.262501  43.750000  41.370625   94640000
2018-01-08  43.587502  43.902500  43.482498  43.587502  41.216961   82271200
```

In [9]:
```
data1 = data.reset_index()
stock_data = data1[['Date','Close']]
```

In [10]:
```
stock_data
```

Out[10]:

	Date	Close
0	2018-01-02	43.064999
1	2018-01-03	43.057499
2	2018-01-04	43.257500
3	2018-01-05	43.750000
4	2018-01-08	43.587502
...	...	...
1254	2022-12-23	131.860001
1255	2022-12-27	130.029999
1256	2022-12-28	126.040001
1257	2022-12-29	129.610001
1258	2022-12-30	129.929993

1259 rows × 2 columns

In [11]:
```
# Calculate the moving average and add it to the new DataFrame
stock_data['STMA'] = stock_data['Close'].rolling(50).mean()

# Calculate the moving average and add it to the new DataFrame
stock_data['LTMA'] = stock_data['Close'].rolling(200).mean()
```

In [ ]:
```
```

In [21]:
```
# Function to implement trading signals with one-day prior check
def implement_trading_signals(stock_data):
    stock_data['Signal'] = 'Hold'  # Initialize a 'Signal' column with 'Hold'
    stock_data['STMA_Prev'] = stock_data['STMA'].shift(1)
    stock_data['LTMA_Prev'] = stock_data['LTMA'].shift(1)

    # Counter for crossovers
    crossover_count = 0
    buy_occurred = False  # Variable to track if a buy signal has occurred

    for i in range(1, len(stock_data)):
        if stock_data['STMA'][i] > stock_data['LTMA'][i] and stock_data['STMA_Prev'
            # Buy Signal: STMA crosses above LTMA (and was below LTMA one day prior
            stock_data.at[i, 'Signal'] = 'Buy'
            crossover_count += 1
            buy_occurred = True
        elif stock_data['STMA'][i] < stock_data['LTMA'][i] and stock_data['STMA_Pre
            # Sell Signal: STMA crosses below LTMA (and was above LTMA one day pric
            if not buy_occurred:
                # If a buy signal has not occurred, set the current sell signal to
                stock_data.at[i, 'Signal'] = 'Hold'
            else:
                # If a buy signal has occurred, set the current sell signal to 'Sel
                stock_data.at[i, 'Signal'] = 'Sell'
                crossover_count += 1

    # Drop the 'STMA_Prev' and 'LTMA_Prev' columns
    stock_data.drop(['STMA_Prev', 'LTMA_Prev'], axis=1, inplace=True)

    print("Total Crossovers:", crossover_count)
    return stock_data
```

In [22]:
```
# Backtesting Strategy with Tabular Output
def backtest_strategy(stock_data, initial_capital=1000000, sell_multiplier=0.5):
    capital = initial_capital
    position = 0
    current_buy = 0
    results = pd.DataFrame(columns=['Date', 'Close Price', 'SMA', 'LMA', 'Signal',
                                    'Capital in Hand', 'Stocks in Hand', 'Returns(i

    for index, row in stock_data.iterrows():
        if row['Signal'] == 'Buy':
            # Buy Signal: Invest all available capital
            current_buy = capital // row['Close']
            capital = capital - (current_buy * row['Close'])
            position += current_buy
            returns = ((capital - initial_capital) + (position * row['Close'])) / i
            total_returns = capital + (position * row['Close'])
            results = results.append({'Date': row["Date"], 'Close Price': row['Clos
                                      'SMA': row['STMA'], 'LMA': row['LTMA'],
                                      'Signal': 'Buy', 'Capital in Hand': capital,
                                      'Stocks in Hand': position, 'Returns(in %)':
                                      'Total Capital': total_returns}, ignore_index
        elif row['Signal'] == 'Sell':
            # Sell Signal: Sell only a fraction of the invested stocks
            sold_position = position * sell_multiplier
            capital += sold_position * row['Close']
            position -= sold_position
            returns = ((capital - initial_capital) + (position * row['Close'])) / i
            total_returns = capital + (position * row['Close'])
            results = results.append({'Date': row["Date"], 'Close Price': row['Clos
                                      'SMA': row['STMA'], 'LMA': row['LTMA'],
                                      'Signal': 'Sell', 'Capital in Hand': capital,
                                      'Stocks in Hand': position, 'Returns(in %)':
                                      'Total Capital': total_returns}, ignore_index

        # Update the 'Position' column in the stock_data DataFrame
        stock_data.at[index, 'Position'] = position

    # Calculate the total portfolio value (capital + stock value)
    stock_data['Portfolio'] = capital + (stock_data['Position'] * stock_data['Close

    # Calculate daily returns
    stock_data['Daily_Return'] = stock_data['Portfolio'].pct_change()

    return results, stock_data
```

In [23]:
```
#Implementing the strategy
stock_data = implement_trading_signals(stock_data)
results, stock_data = backtest_strategy(stock_data, initial_capital=1000000, sell_m
```

```
Total Crossovers: 4
```

In [24]:
```
# Display tabular results
print(results)
```

```
        Date  Close Price         SMA         LMA Signal  Capital in Hand  \
0 2019-05-07    50.715000   48.125600   48.102713    Buy     1.626991e+00
1 2022-06-03   145.380005  159.089801  159.486850   Sell     1.433303e+06
2 2022-09-28   149.839996  160.288400  160.208600    Buy     8.353016e+01
3 2022-09-30   138.199997  159.734200  159.861650   Sell     1.342282e+06

   Stocks in Hand  Returns(in %)  Total Capital
0         19718.0       0.000000   1.000000e+06
1          9859.0     186.660456   2.866605e+06
2         19424.0     191.057562   2.910576e+06
3          9712.0     168.448027   2.684480e+06
```

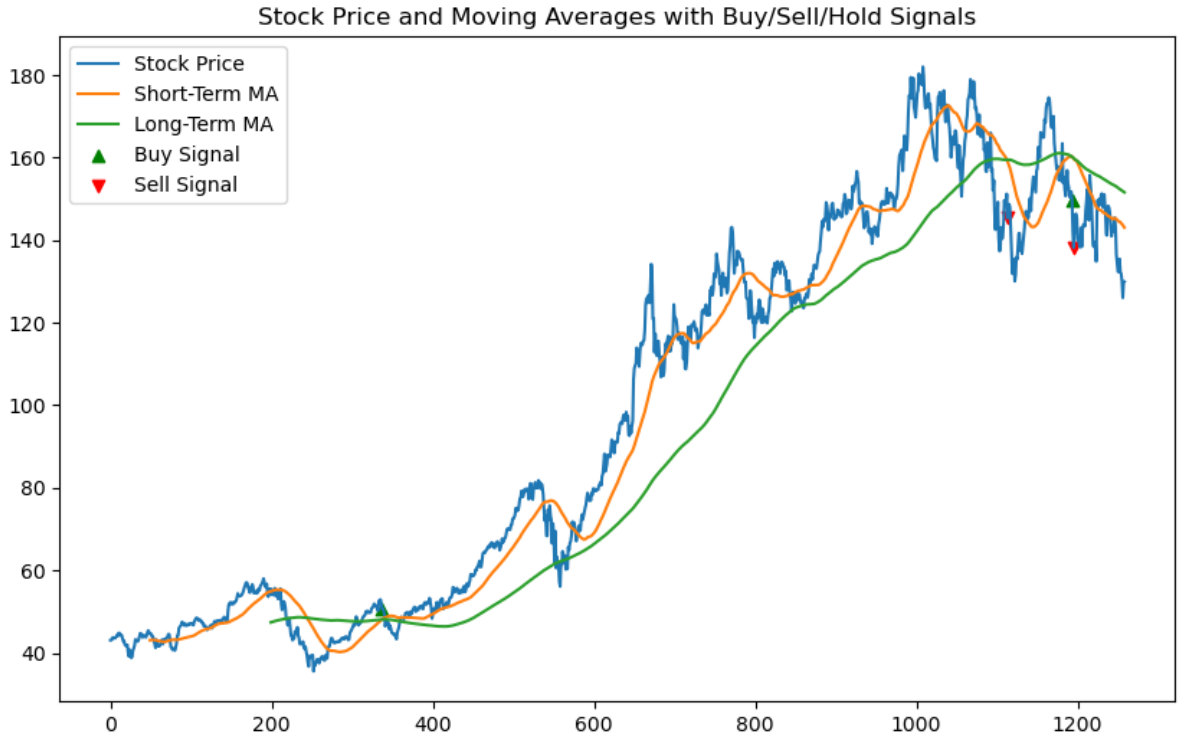
In [25]:

```
# Plotting the results
plt.figure(figsize=(10, 6))
plt.plot(stock_data['Close'], label='Stock Price')
plt.plot(stock_data['STMA'], label='Short-Term MA')
plt.plot(stock_data['LTMA'], label='Long-Term MA')
plt.title('Stock Price and Moving Averages with Buy/Sell/Hold Signals')

# Use the index for both x and y values in scatter plot
buy_signals = stock_data[stock_data['Signal'] == 'Buy']
sell_signals = stock_data[stock_data['Signal'] == 'Sell']
hold_signals = stock_data[stock_data['Signal'] == 'Hold']

# Scatter plot for Buy and Sell signals
plt.scatter(buy_signals.index, buy_signals['Close'], marker='^', color='g', label='
plt.scatter(sell_signals.index, sell_signals['Close'], marker='v', color='r', label

plt.legend()
plt.show()
```



In [ ]: