Data Pull -> Using previous volatility -> Training GARCH model -> Predicting volatitliy for next 500 days

In [42]:

```
#Libraries and fetching data

import yfinance as yf
import math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from datetime import date

# Fetching data since 2010 till today
ticker= "TSLA"
start_date = "2010-01-01"
end_date = date.today().strftime("%Y-%m-%d")
tsla_data = yf.download(ticker, start=start_date, end=end_date)
```

```
[*********************100%***********************]  1 of 1 completed
```

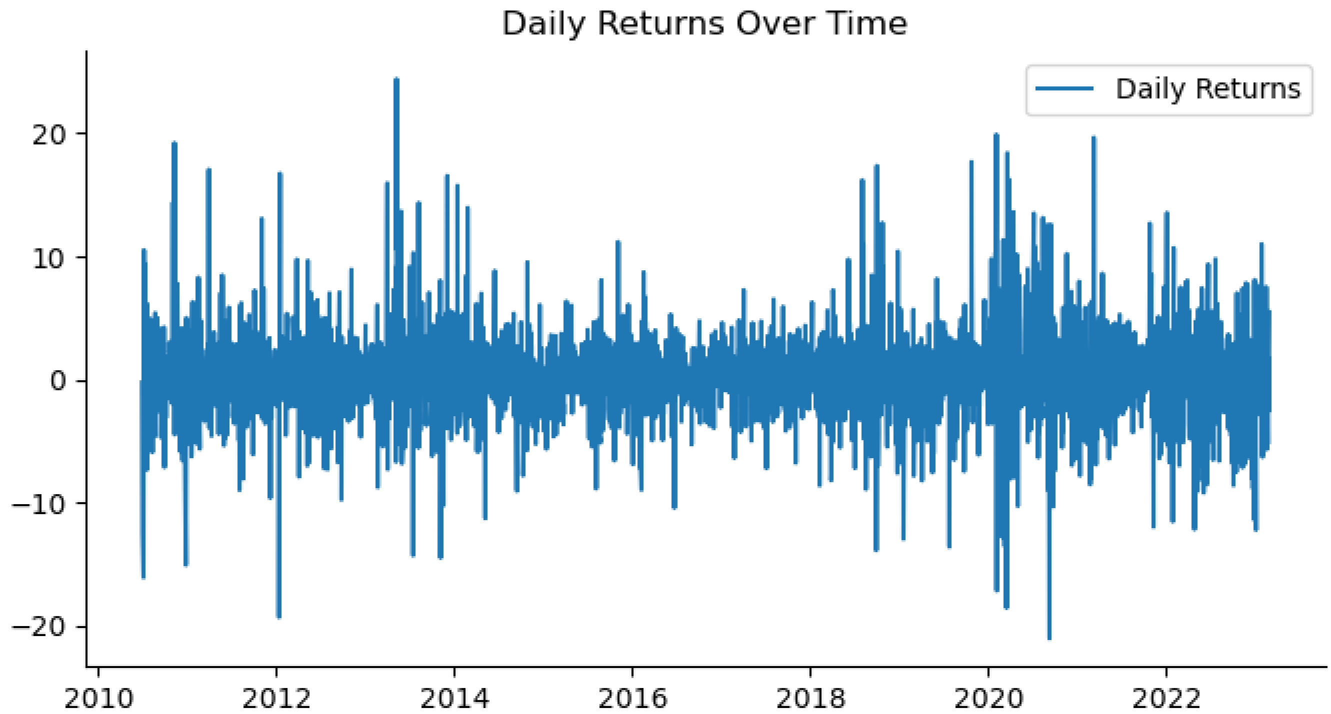
In [ ]:

Daily, Monthly and Annual Volatility!

In [43]:

```
tsla_data['Return'] = 100 * (tsla_data['Close'].pct_change())
tsla_data.dropna(inplace=True)

fig,ax = plt.subplots(figsize=(8,4))
ax.spines[['top','right']].set_visible(False)
plt.plot(tsla_data['Return'], label = 'Daily Returns')
plt.legend(loc='upper right')
plt.title('Daily Returns Over Time')

daily_volatility = tsla_data['Return'].std()
monthly_volatility = math.sqrt(21) * daily_volatility
annual_volatility = math.sqrt(252) * daily_volatility

from tabulate import tabulate
print(tabulate([['Tesla', daily_volatility, monthly_volatility, annual_volatility]],
               headers=['Daily Volatility %', 'Monthly Volatility %', 'Annual Volatility
               tablefmt='fancy_grid', stralign='center', numalign='center', floatfmt=".2
```

	Daily Volatility %	Monthly Volatility %	Annual Volatility %
Tesla	3.62	16.58	57.43



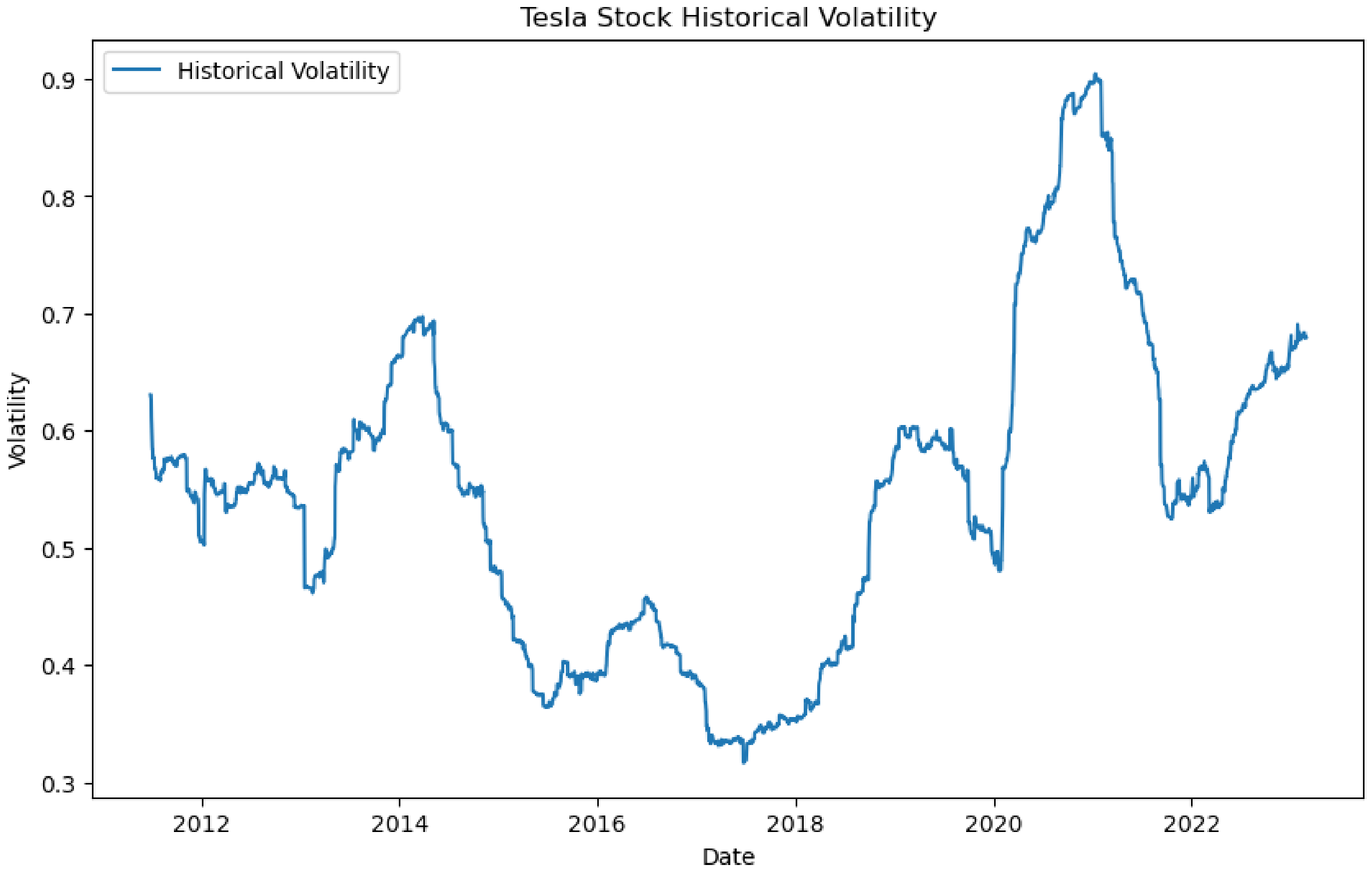
The below data shows: mean return (mu) = 17.1%, long term average voalitility (omega) = 13%, short-run voalititly (alpha) = 3%, persistence of volatility (beta) = 95.7%

In [45]:
```
# Fitting GARCH model and forecasting volatility till 2024 September 20th
from arch import arch_model
from arch.__future__ import reindexing

garch_model = arch_model(tsla_data['Return'], p=1, q=1, mean='constant', vol='GARCH', di
gm_result = garch_model.fit(disp='off')
print(gm_result.params)
print('\n')
```

```
mu          0.171850
omega       0.130736
alpha[1]    0.032224
beta[1]     0.957707
Name: params, dtype: float64
```

Plot the historical volatility of the Tesla stock

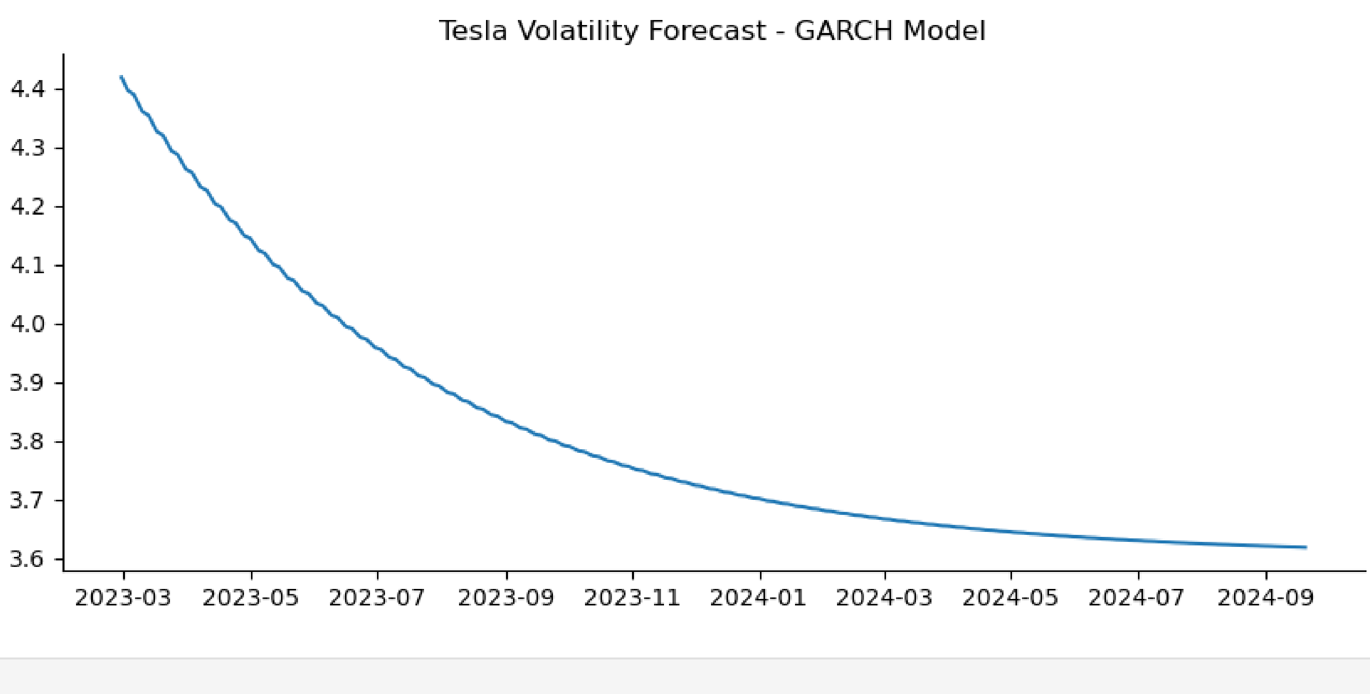
In [51]:
```
plt.figure(figsize=(10, 6))
plt.plot(tesla['Date'], tesla['Volatility'], label='Historical Volatility')
plt.title('Tesla Stock Historical Volatility')
plt.xlabel('Date')
plt.ylabel('Volatility')
plt.legend()
plt.show()
```



In [ ]:

In [46]:
```
start_date = tsla_data.index[-1].date() + pd.Timedelta(days=1)
end_date = "2024-09-20"
forecast_horizon = pd.date_range(start_date, end_date, freq='B')
gm_forecast = gm_result.forecast(horizon=len(forecast_horizon), start=start_date)
forecast = pd.DataFrame(np.sqrt(gm_forecast.variance.values).T, index=forecast_horizon,

fig, ax = plt.subplots(figsize=(10, 4))
ax.spines[['top','right']].set_visible(False)
plt.plot(forecast)
plt.title('Tesla Volatility Forecast - GARCH Model')
```

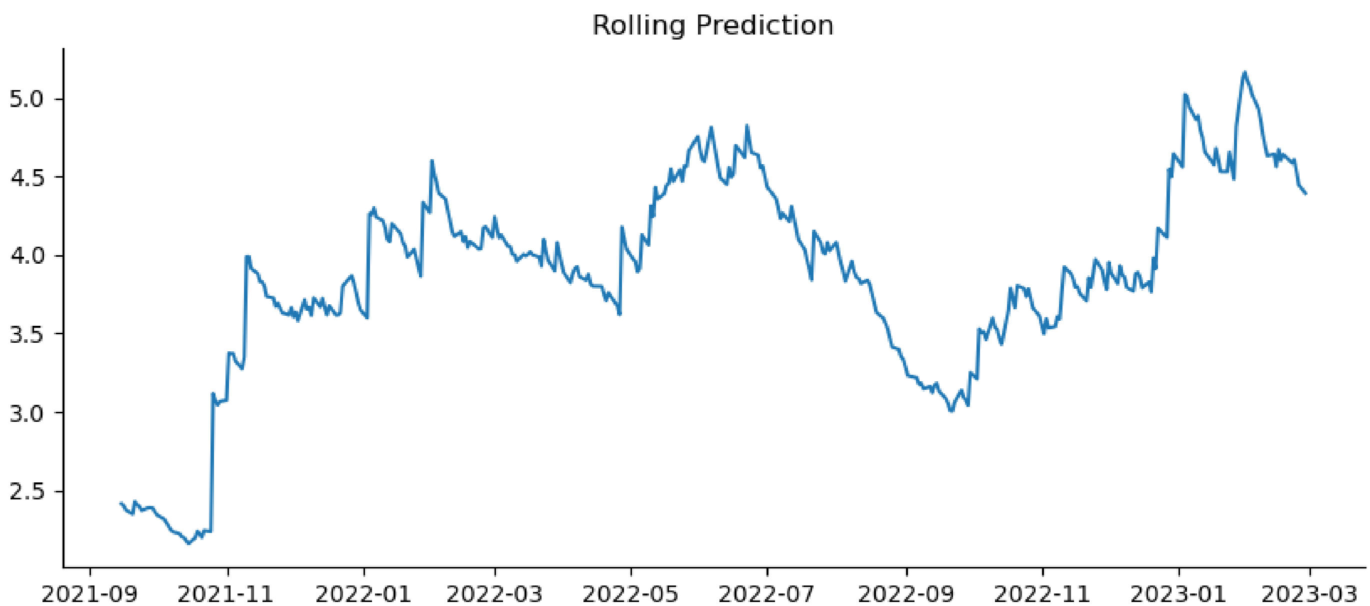
Out[46]:
```
Text(0.5, 1.0, 'Tesla Volatility Forecast - GARCH Model')
```



In [ ]:

In [47]:
```
# Rolling prediction
rolling_predictions = []
test_size = 365

for i in range(test_size):
    train = tsla_data['Return'][:-(test_size-i)]
    model = arch_model(train, p=1, q=1)
    model_fit = model.fit(disp='off')
    pred = model_fit.forecast(horizon=1)
    rolling_predictions.append(np.sqrt(pred.variance.values[-1,:][0]))

rolling_predictions = pd.Series(rolling_predictions, index=tsla_data['Return'].index[-36

fig, ax = plt.subplots(figsize=(10,4))
ax.spines[['top','right']].set_visible(False)
plt.plot(rolling_predictions)
plt.title('Rolling Prediction')
```

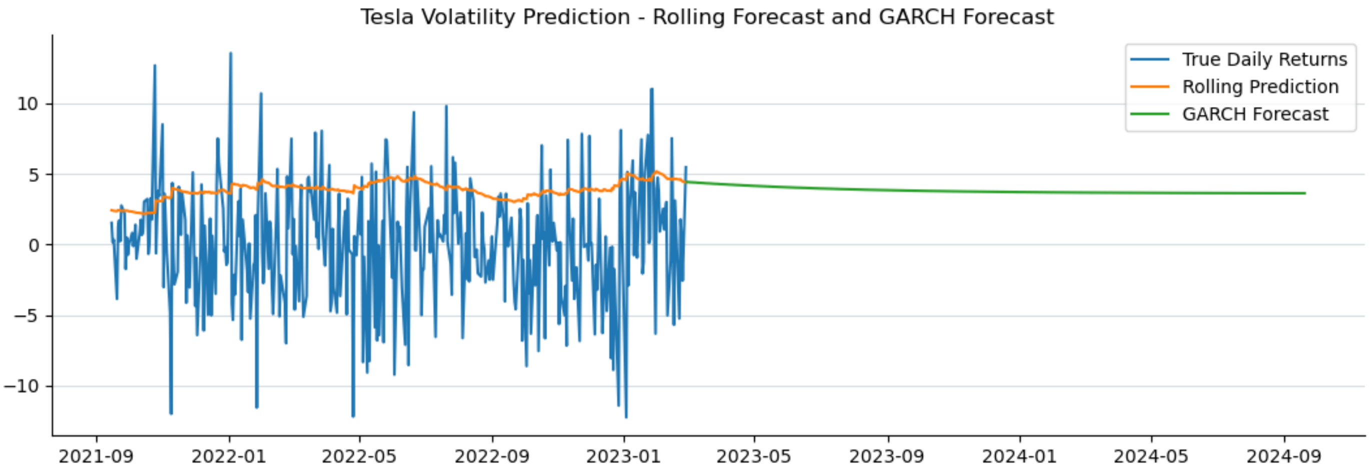
Out[47]:
```
Text(0.5, 1.0, 'Rolling Prediction')
```



Below is the forecast using GARCH

In [48]:
```
fig, ax = plt.subplots(figsize=(13, 4))
ax.grid(which="major", axis='y', color='#758D99', alpha=0.3, zorder=1)
ax.spines[['top','right']].set_visible(False)
plt.plot(tsla_data['Return'][-365:])
plt.plot(rolling_predictions)
plt.plot(forecast)
plt.title('Tesla Volatility Prediction - Rolling Forecast and GARCH Forecast')
plt.legend(['True Daily Returns', 'Rolling Prediction', 'GARCH Forecast'])
```

Out[48]:
```
<matplotlib.legend.Legend at 0x7fbbd2099790>
```



In [64]:
```
annual_volatility = daily_volatility * np.sqrt(252)
print("Projected annual volatility:", annual_volatility)
print('The annualized volatility from today until 20th September 2024 is:', round(annual
```

```
Projected annual volatility: 57.427022111834475
The annualized volatility from today until 20th September 2024 is: 4.6
```

The Projected annual volatiltiy is : 57.43%

The predicted daily volatility value at the end of the forecast horizon (20th September 2024) is: 3.61

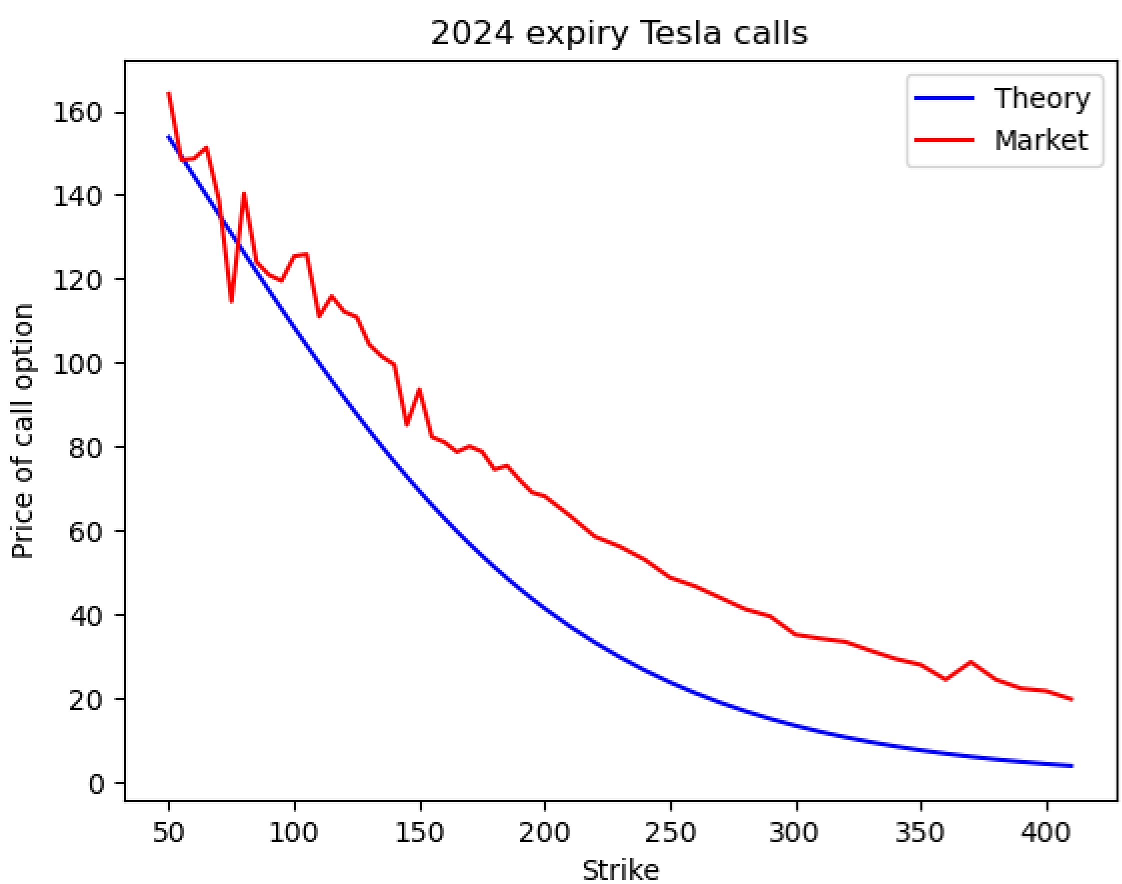
Using Volatility forecasted from GARCH into the BSM model, Rate = Fed 1 year yeild, Time - 20th Sept, 2024, Stock = 205

In [43]:
```
import matplotlib.pyplot as plt
import yfinance as yf
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt

# Fetch Tesla stock data from Yahoo Finance API
tesla = yf.Ticker("TSLA")
tesla_history = tesla.history(start="2010-01-01", end="2023-03-01")
```

Theory vs market plot with historical values

In [59]:
```
```

Out[59]: Text(0, 0.5, 'Price of call option')



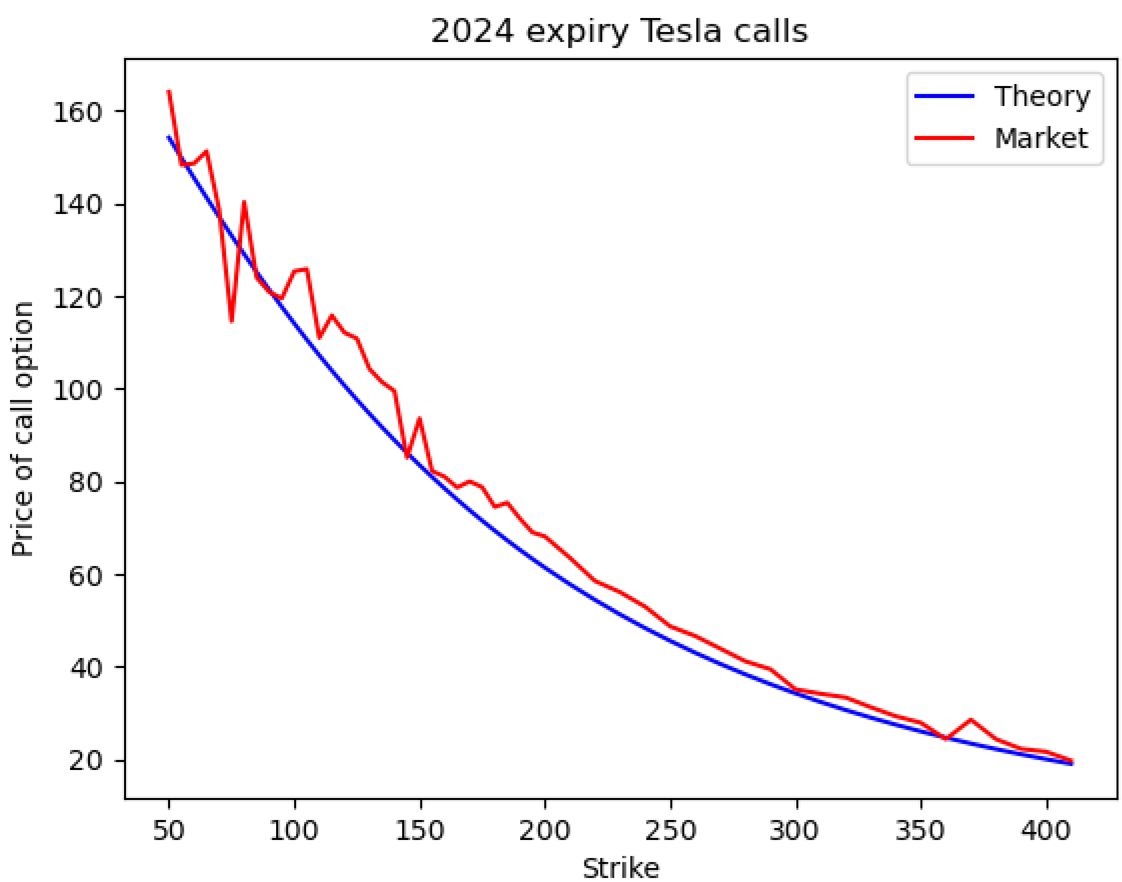
AFTER feeding GARCH/ forecasted volatility

In [58]:
```
# Define Black-Scholes function for European call options
def bs_call(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + sigma ** 2 / 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    return S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)

data = tesla_calls
```

```python
# Define parameters
S = 200    # Stock price
r = 0.050  # Risk-free rate
sigma = 0.57  # Volatility # Can use the forecasted from GARCH model

# Calculate option price for each row using Black-Scholes
optionPrices = []
for index, row in data.iterrows():
    K = row['strike']
    T = 570 / 365
    optionPrices.append(bs_call(S, K, T, r, sigma))

# Add Theoretical optionPrice column
data['optionPrice'] = optionPrices

plt.plot(data['strike'],data['optionPrice'], color = 'blue', label = "Theory")
plt.plot(data['strike'],data['lastPrice'], color = 'red', label = "Market")

plt.legend()
plt.title("2024 expiry Tesla calls")
plt.xlabel('Strike')
plt.ylabel('Price of call option')
```

Out[58]:

```
Text(0, 0.5, 'Price of call option')
```



In [ ]:

In [ ]:

```
## Prophet Model to predict fore
```

In [ ]:

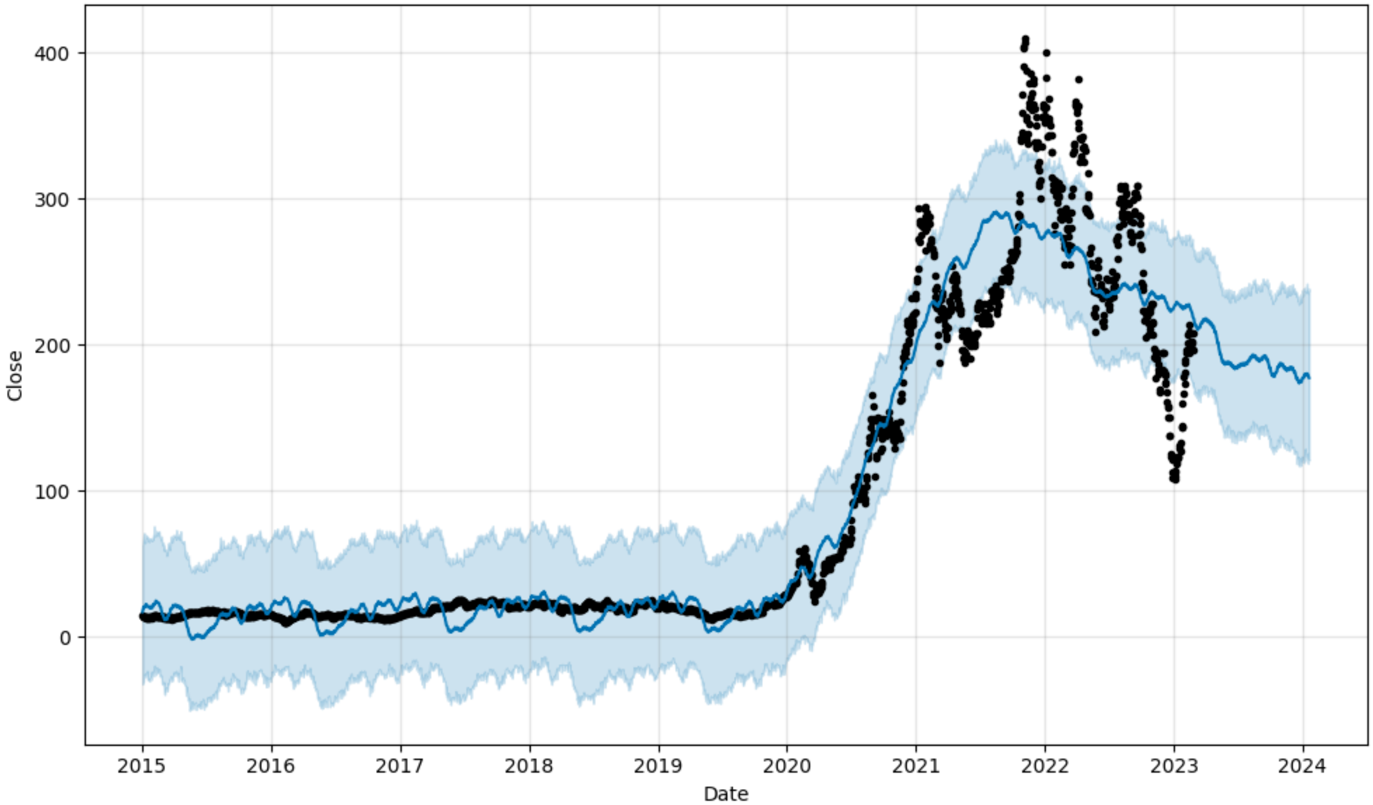
```
```

In [16]:

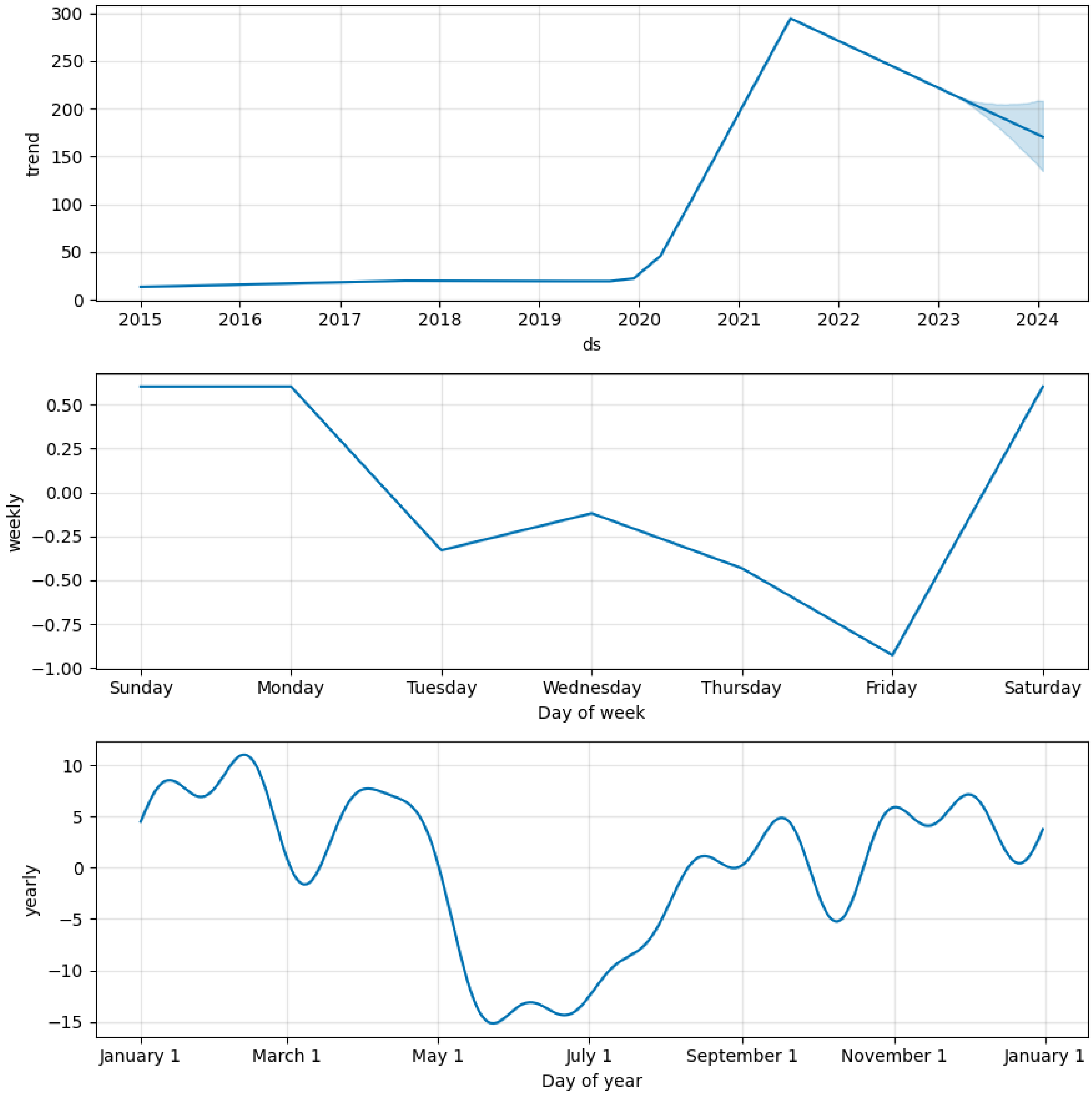
```
import pandas as pd
from prophet import Prophet
import matplotlib.pyplot as plt

# load the data
df = yf.download('TSLA', start='2010-06-29', end='2023-02-28')
tesla_data.index = pd.to_datetime(tesla_data.index)

df = df[['Close']].reset_index()
df.columns = ['ds', 'y']
df = df[df['ds'] >= '2015-01-01'] # use data from 2015

# create and fit the Prophet model
model = Prophet(interval_width=0.90)
model.fit(df)

# create a dataframe with future dates
future_dates = model.make_future_dataframe(periods=1081, freq='D', include_history=True)

# filter future dates to predict only till 2024-01-19
future_dates = future_dates[future_dates['ds'] <= '2024-01-19']

# make predictions for the future dates
forecast = model.predict(future_dates)

# plot the forecast
model.plot(forecast, xlabel='Date', ylabel='Close')
model.plot_components(forecast)

plt.show()
```

```
[*********************100%***********************]  1 of 1 completed
```

```
23:23:17 - cmdstanpy - INFO - Chain [1] start processing
23:23:17 - cmdstanpy - INFO - Chain [1] done processing
```



400
300
200
100
0
Close
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
Date



In [ ]:

In [17]:
```
last_prediction = forecast.iloc[-1]
lower_bound = last_prediction['yhat_lower']
upper_bound = last_prediction['yhat_upper']

last_prediction
```

Out[17]:
```
ds                            2024-01-19 00:00:00
trend                                  170.455599
yhat_lower                             121.092149
yhat_upper                             238.454923
trend_lower                            134.748816
trend_upper                            208.356281
additive_terms                           6.918777
additive_terms_lower                     6.918777
additive_terms_upper                     6.918777
weekly                                  -0.926789
weekly_lower                            -0.926789
weekly_upper                            -0.926789
yearly                                   7.845566
```

```
yearly_lower                                7.845566
yearly_upper                                7.845566
multiplicative_terms                             0.0
multiplicative_terms_lower                       0.0
multiplicative_terms_upper                       0.0
yhat                                      177.374376
Name: 2377, dtype: object
```

90 % confidence interval 122 to 238, with mean prediction at 178

In [ ]:

In [ ]:

In [ ]: