In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.stats.diagnostic import acorr_ljungbox
import numpy as np
from sklearn.metrics import mean_squared_error
```

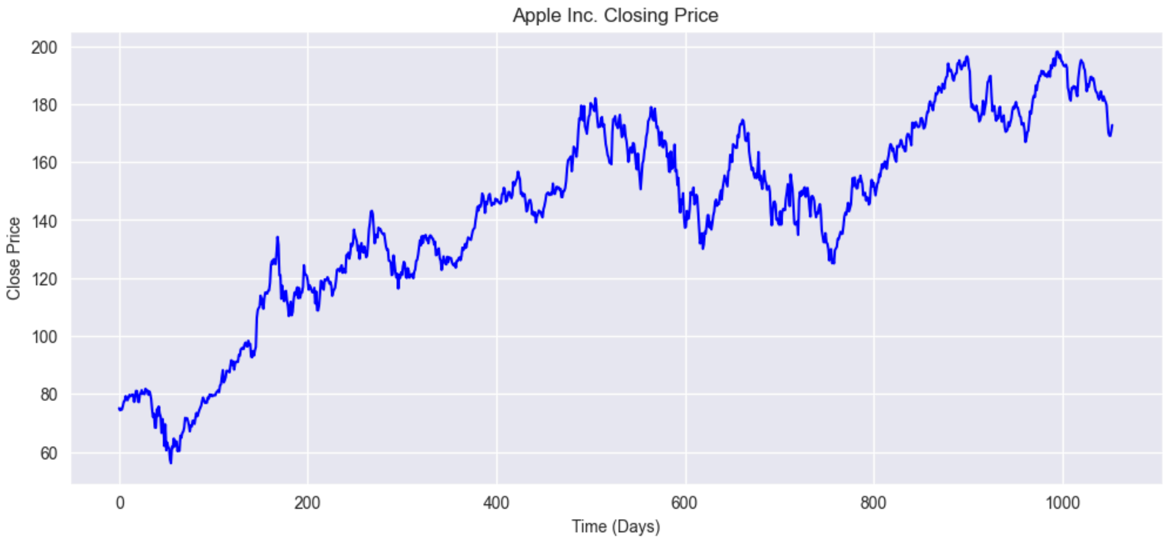
In [5]:

```
df = pd.read_csv("dataset\Stock_price.csv")
prices = df['Close']  # Use Close price for modeling

print("First 5 rows:\n", df.head())
```

```
First 5 rows:
         Open       High        Low      Close  Adj Close     Volume
0  74.059998  75.150002  73.797501  75.087502  73.059425  135480400
1  74.287498  75.144997  74.125000  74.357498  72.349144  146322800
2  73.447502  74.989998  73.187500  74.949997  72.925636  118387200
3  74.959999  75.224998  74.370003  74.597504  72.582649  108872000
4  74.290001  76.110001  74.290001  75.797501  73.750244  132079200
```

In [7]:

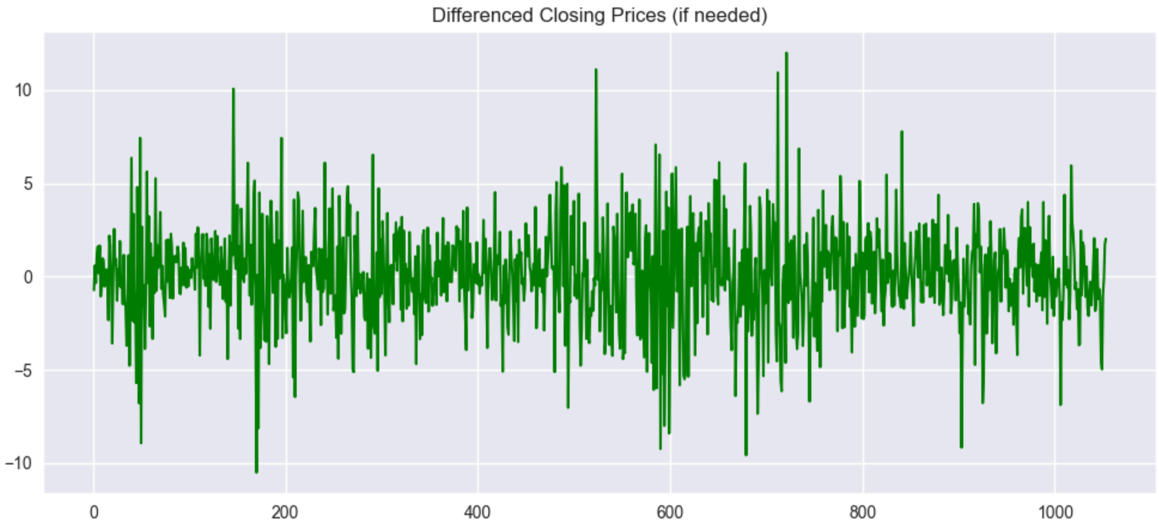
```
# Set Seaborn style for plots
sns.set_style('darkgrid')

# 3. Visualize Closing Prices
plt.figure(figsize=(12,5))
plt.plot(prices, color='blue')
plt.title('Apple Inc. Closing Price')
plt.xlabel('Time (Days)')
plt.ylabel('Close Price')
plt.show()
```



In [8]:

```
# 4. Check Stationarity
adf_result = adfuller(prices)
print('ADF Statistic: %.4f' % adf_result[0])
print('p-value: %.4f' % adf_result[1])

if adf_result[1] > 0.05:
    prices_diff = prices.diff().dropna()
    print("Series is non-stationary → Differenced series created")
else:
    prices_diff = prices
    print("Series is stationary → Use original series")

# Plot differenced series
plt.figure(figsize=(12,5))
plt.plot(prices_diff, color='green')
plt.title('Differenced Closing Prices (if needed)')
plt.show()
```

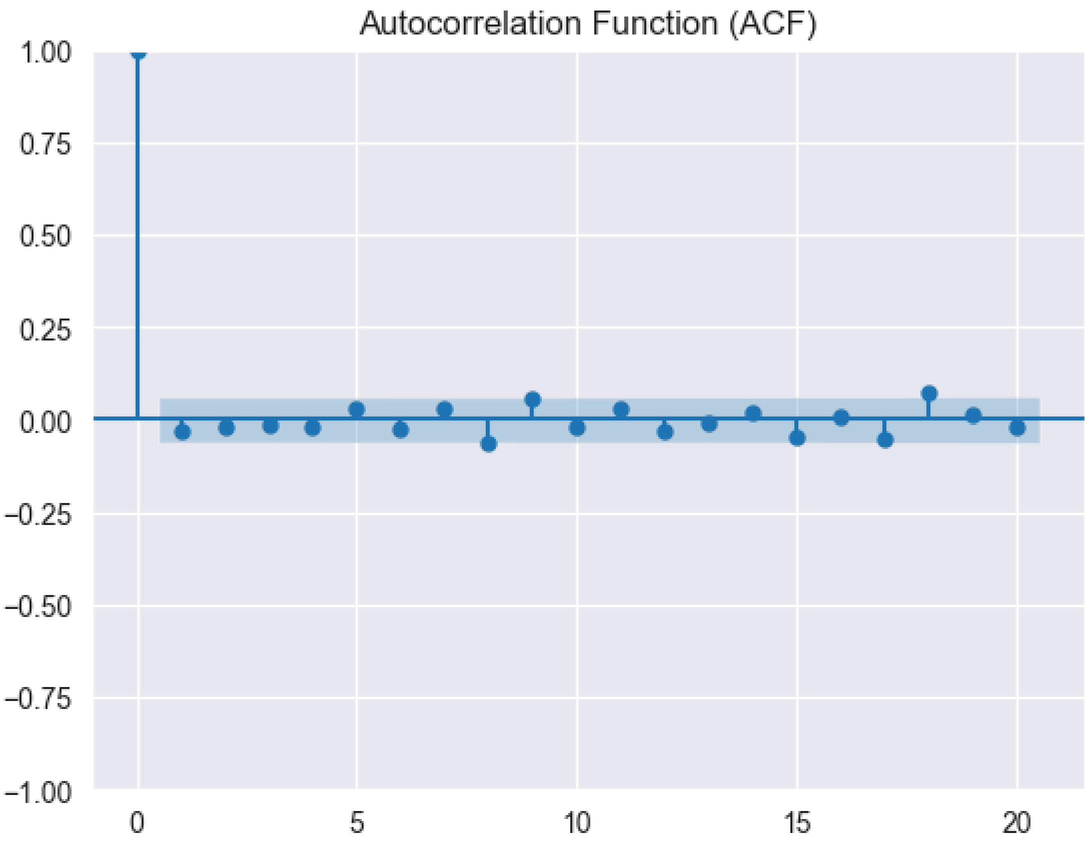
```
ADF Statistic: -1.9040
p-value: 0.3302
Series is non-stationary → Differenced series created
```


Differenced Closing Prices (if needed)

In [9]:

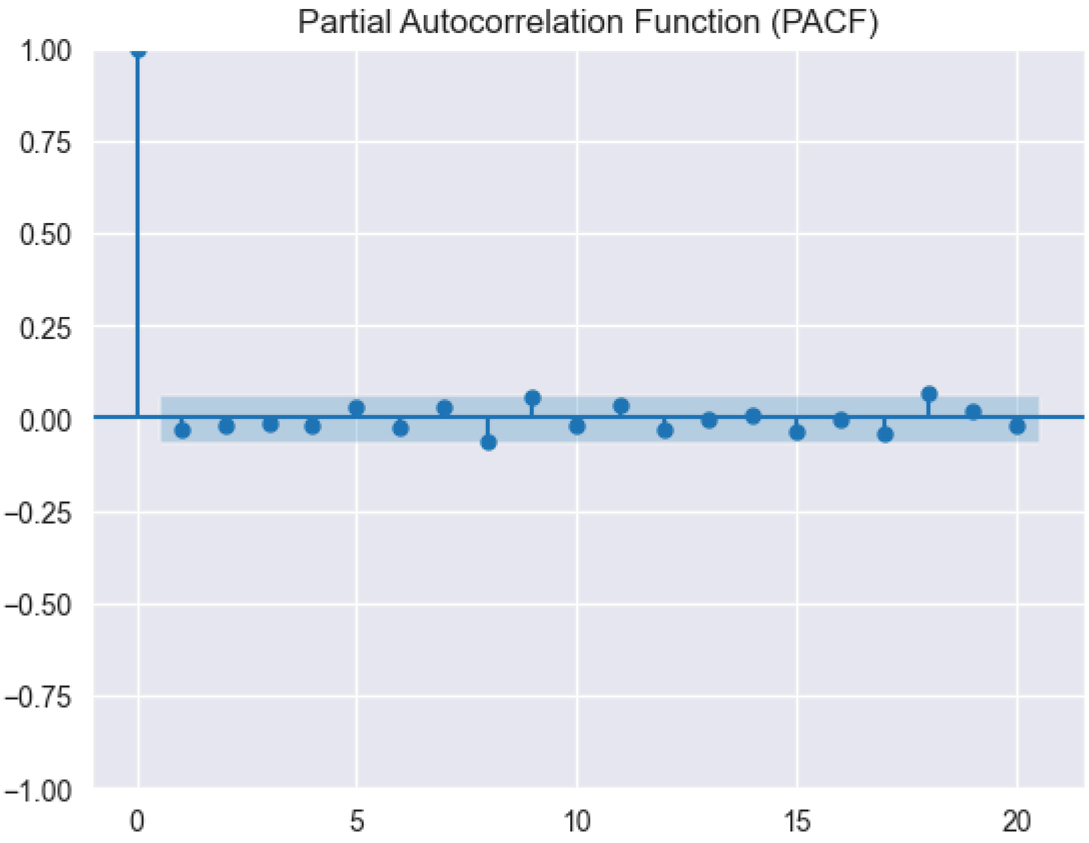
```
# 5. ACF and PACF
plt.figure(figsize=(12,5))
plot_acf(prices_diff, lags=20)
plt.title('Autocorrelation Function (ACF)')
plt.show()

plt.figure(figsize=(12,5))
plot_pacf(prices_diff, lags=20)
plt.title('Partial Autocorrelation Function (PACF)')
plt.show()
```

```
<Figure size 1200x500 with 0 Axes>
```



Autocorrelation Function (ACF)

```
<Figure size 1200x500 with 0 Axes>
```



Partial Autocorrelation Function (PACF)

In [10]:

```
# 6. Fit AR Model
ar_lag = 3  # choose from PACF plot
ar_model = AutoReg(prices_diff, lags=ar_lag).fit()
print("\nAR Model Summary:\n", ar_model.summary())
```

```
AR Model Summary:
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1053
Model:                     AutoReg(3)   Log Likelihood               -2506.880
Method:               Conditional MLE   S.D. of innovations              2.634
Date:                Sat, 07 Feb 2026   AIC                           5023.761
Time:                        15:07:22   BIC                           5048.544
Sample:                             3   HQIC                          5033.158
                                 1053
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0989      0.081      1.214      0.225      -0.061       0.259
Close.L1      -0.0287      0.031     -0.928      0.353      -0.089       0.032
Close.L2      -0.0187      0.031     -0.604      0.546      -0.079       0.042
Close.L3      -0.0125      0.031     -0.404      0.687      -0.073       0.048
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.5894           -3.8261j            4.1431           -0.1873
AR.2            1.5894           +3.8261j            4.1431            0.1873
AR.3           -4.6763           -0.0000j            4.6763           -0.5000
-----------------------------------------------------------------------------
```

```
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result,
forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
```

In [11]:

```
# 7. Fit MA Model
ma_order = 2  # choose from ACF plot
ma_model = ARIMA(prices_diff, order=(0,0,ma_order)).fit()
print("\nMA Model Summary:\n", ma_model.summary())
```

```
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result,
forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result,
forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result,
forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
```

```
MA Model Summary:
                                SARIMAX Results                                
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1053
Model:                 ARIMA(0, 0, 2)   Log Likelihood               -2512.684
Date:                Sat, 07 Feb 2026   AIC                           5033.367
Time:                        15:07:33   BIC                           5053.205
Sample:                             0   HQIC                          5040.888
                               - 1053                                         
Covariance Type:                  opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0927      0.079      1.178      0.239      -0.062       0.247
ma.L1         -0.0290      0.026     -1.103      0.270      -0.080       0.022
ma.L2         -0.0187      0.027     -0.685      0.494      -0.072       0.035
sigma2         6.9209      0.228     30.394      0.000       6.475       7.367
=================================================================================
==
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               119.
26
Prob(Q):                              1.00   Prob(JB):                         0.
00
Heteroskedasticity (H):               0.94   Skew:                            -0.
08
Prob(H) (two-sided):                  0.53   Kurtosis:                         4.
64
=================================================================================
==

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-st
ep).
```

In [12]:

```python
# 8. Fit ARMA Model (AR + MA)
arma_model = ARIMA(prices_diff, order=(ar_lag,0,ma_order)).fit()
print("\nARMA Model Summary:\n", arma_model.summary())
```

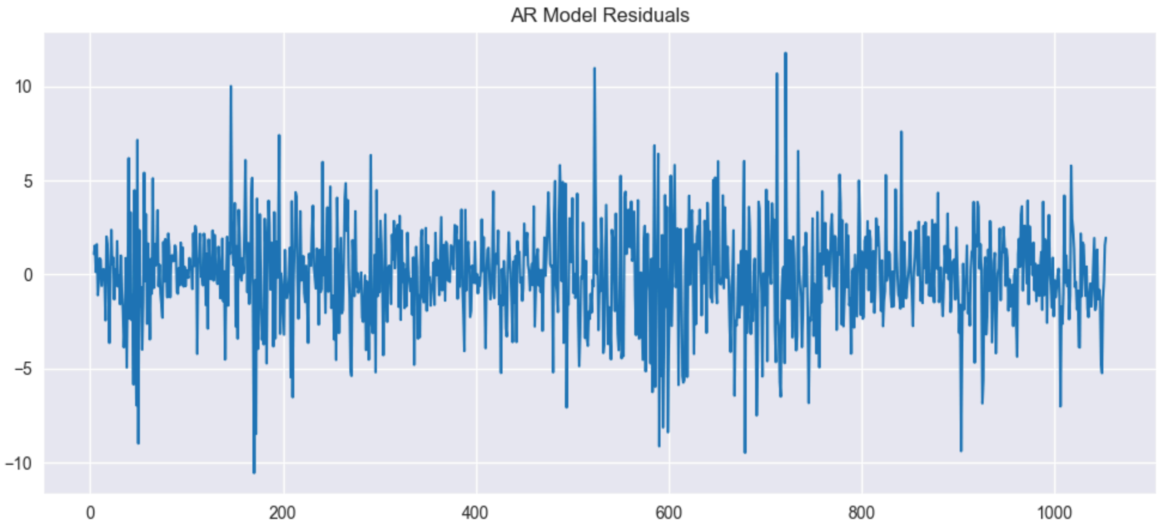
```
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result,
forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result,
forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:473: ValueWarning: An unsupported index was provided. As a result,
forecasts cannot be generated. To use the model for forecasting, use one of the s
upported classes of index.
  self._init_dates(dates, freq)
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\sta
tespace\sarimax.py:966: UserWarning: Non-stationary starting autoregressive param
eters found. Using zeros as starting parameters.
  warn('Non-stationary starting autoregressive parameters'
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\sta
tespace\sarimax.py:978: UserWarning: Non-invertible starting MA parameters found.
Using zeros as starting parameters.
  warn('Non-invertible starting MA parameters found.'
```

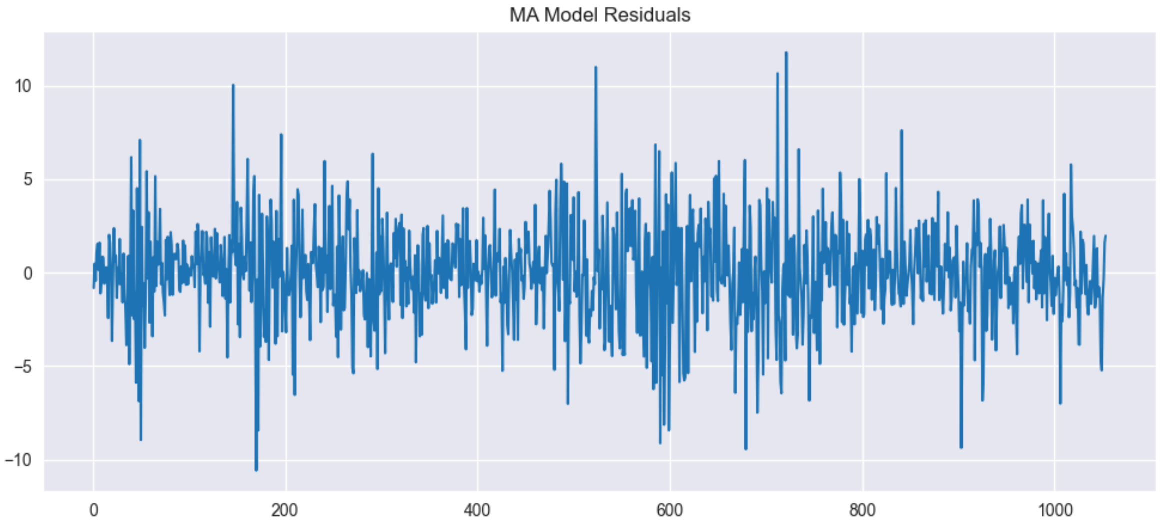
```
ARMA Model Summary:
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1053
Model:                 ARIMA(3, 0, 2)   Log Likelihood               -2510.795
Date:                Sat, 07 Feb 2026   AIC                           5035.591
Time:                        15:07:45   BIC                           5070.306
Sample:                             0   HQIC                          5048.752
                               - 1053
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0764      0.079      0.964      0.335      -0.079       0.232
ar.L1          0.3515      0.060      5.861      0.000       0.234       0.469
ar.L2         -0.9516      0.047    -20.264      0.000      -1.044      -0.860
ar.L3         -0.0314      0.028     -1.112      0.266      -0.087       0.024
ma.L1         -0.3713      0.051     -7.265      0.000      -0.472      -0.271
ma.L2          0.9563      0.047     20.133      0.000       0.863       1.049
sigma2         6.8183      0.226     30.177      0.000       6.375       7.261
=================================================================================
==
Ljung-Box (L1) (Q):                   0.10   Jarque-Bera (JB):               121.
29
Prob(Q):                              0.76   Prob(JB):                         0.
00
Heteroskedasticity (H):               0.93   Skew:                            -0.
07
Prob(H) (two-sided):                  0.49   Kurtosis:                         4.
66
=================================================================================
==

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-st
ep).
```

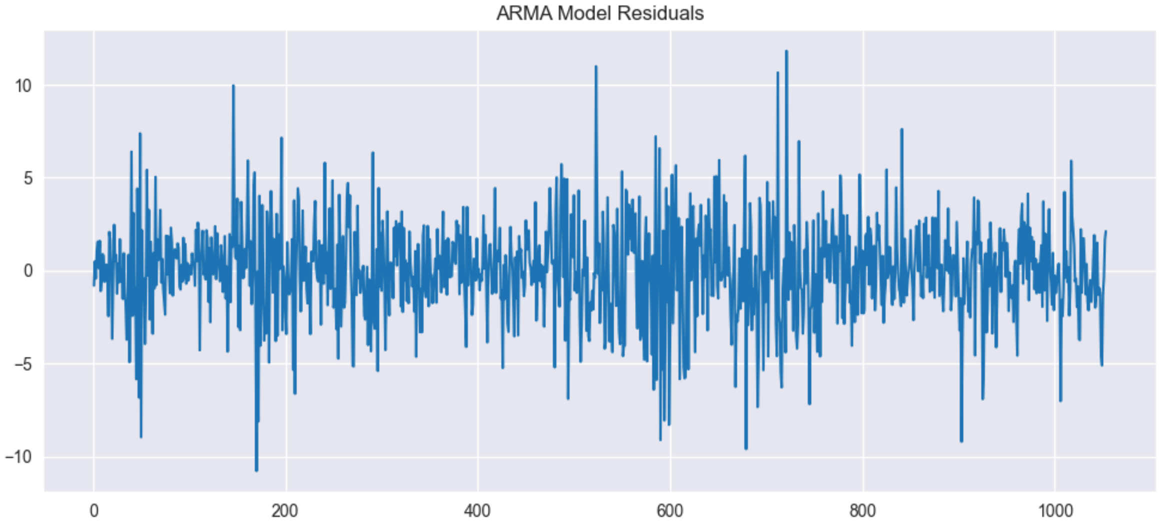
```
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\base\mo
del.py:607: ConvergenceWarning: Maximum Likelihood optimization failed to converg
e. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [14]:

```
# 9. Residual Analysis
plt.figure(figsize=(12,5))
plt.plot(ar_model.resid)
plt.title('AR Model Residuals')
plt.show()

plt.figure(figsize=(12,5))
plt.plot(ma_model.resid)
plt.title('MA Model Residuals')
plt.show()

plt.figure(figsize=(12,5))
plt.plot(arma_model.resid)
plt.title('ARMA Model Residuals')
plt.show()

# Ljung-Box test for ARMA residuals
lb_test = acorr_ljungbox(arma_model.resid, lags=[10], return_df=True)
print("\nLjung-Box Test for ARMA Residuals:\n", lb_test)
```







```
Ljung-Box Test for ARMA Residuals:
      lb_stat  lb_pvalue
10  9.302062   0.503697
```

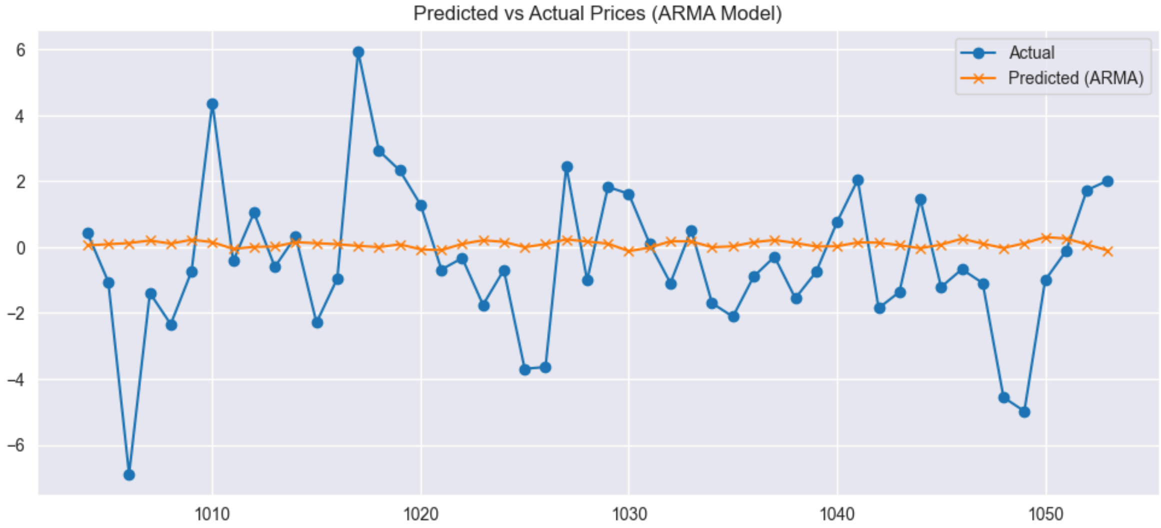
In [15]:
```
# 10. Model Comparison
print("\nAR Model AIC:", ar_model.aic)
print("AR Model BIC:", ar_model.bic)
print("\nMA Model AIC:", ma_model.aic)
print("MA Model BIC:", ma_model.bic)
print("\nARMA Model AIC:", arma_model.aic)
print("ARMA Model BIC:", arma_model.bic)
```

```
AR Model AIC: 5023.760851323028
AR Model BIC: 5048.543578538785

MA Model AIC: 5033.367301699888
MA Model BIC: 5053.204895748424

ARMA Model AIC: 5035.590678164631
ARMA Model BIC: 5070.306467749569
```

In [16]:
```
# 11. Forecasting (next 10 days) and Plotting Predicted vs Actual
forecast_steps = 10
arma_forecast = arma_model.forecast(steps=forecast_steps)
print("\nNext", forecast_steps, "days ARMA forecast:\n", arma_forecast)

# Predicted vs Actual (last 50 points)
pred_start = -50
plt.figure(figsize=(12,5))
plt.plot(prices_diff[pred_start:], label='Actual', marker='o')
plt.plot(arma_model.predict(start=len(prices_diff)+pred_start, end=len(prices_di
plt.title('Predicted vs Actual Prices (ARMA Model)')
plt.legend()
plt.show()
```

```
Next 10 days ARMA forecast:
 1053   -0.022011
1054    0.167063
1055    0.140818
1056    0.015851
1057   -0.009045
1058    0.101952
1059    0.168590
1060    0.087167
1061   -0.008358
1062    0.033453
Name: predicted_mean, dtype: float64
```

```
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:837: ValueWarning: No supported index is available. Prediction res
ults will be given with an integer index beginning at `start`.
  return get_prediction_index(
C:\Users\jamiy\AppData\Roaming\Python\Python310\site-packages\statsmodels\tsa\bas
e\tsa_model.py:837: FutureWarning: No supported index is available. In the next v
ersion, calling this method in a model without a supported index will result in a
n exception.
  return get_prediction_index(
```



In [17]:

```
# 12. Calculate RMSE for last 50 points
actual = prices_diff[pred_start:]
predicted = arma_model.predict(start=len(prices_diff)+pred_start, end=len(prices
rmse = np.sqrt(mean_squared_error(actual, predicted))
print("RMSE of ARMA model for last 50 points:", round(rmse,4))
```

```
RMSE of ARMA model for last 50 points: 2.3142
```

In [ ]: