Berkshire Hathaway Stock Price Prediction

Goal:

This machine learning project aims to enhance the accuracy of stock price prediction by applying and improving time series forecasting models on historical data from Berkshire Hathaway.

Background:

Our midterm project was centered around getting the building blocks set. We used a variety of models like ARIMA, Linear Regression, and Random Forests. After some discussion and analysis that is detailed below, our final project combines a variant of ARIMA called SARIMA with Random Forest to create an ensemble for the most accurate results.

Import Libraries and Handle Data

First, let's start off by importing the necessary libraries and loading in our dataset

In [33]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose
from pmdarima.arima import auto_arima
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error
import math

# Load and parse stock data
df = pd.read_csv('./berkshire_hathaway_data.csv', index_col='Date', parse_dates=Tru
```

Plot Feature (Close Price)

We decided to focus in on the closing price because it reflects the final price of the stock at the ending of the trading day. Unlike the high's and low's that can introduce more noise into our model, the close price is often considered the most stable which serves as a good indicator for how the stock performs overall

In [89]:
```
df_close = df['Close']
```

```
#plot close price
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Date')
plt.ylabel('Close Prices')
plt.plot(df_close)
plt.title('Berkshire Hathaway Closing Price')
plt.show()
```

Data Exploration

While analyzing the graph of Berkshire Hathaway's closing prices (2015–2025), we observed the following key trends:

1. Long-Term Upward Trend

The stock demonstrates a strong and consistent upward trajectory over the 10-year period. This suggests sustained investor confidence and reflects the company’s solid portfolio performance and strategic growth over time.

2. The 2020 Dip

Like many companies, Berkshire Hathaway experienced a noticeable drop in early 2020, corresponding to the global COVID-19 market crash. However, the stock rebounded quickly, highlighting the company's resilience and stability amidst economic uncertainty.

3. Post-2021 Acceleration

Following the 2020 recovery, the stock's growth rate noticeably steepens after 2021. This surge indicates strong bullish sentiment and may reflect increased investor optimism, improved economic conditions, or exceptional performance of Berkshire's holdings.

Insights

1. Time Span:

The dataset spans from 2015 to 2025, offering over 2,400 daily data points

2. Growth Trend:

The stock is generally growing upwards

3. Volatility:

Noticeable price drops when major events in the world happen (Covid) and early 2023 witnessed the fall of Silicon Valley Bank and Signature Bank causing investors to be weary.

4. Stability:

Relatively stable outside of certain uncontrollable events.

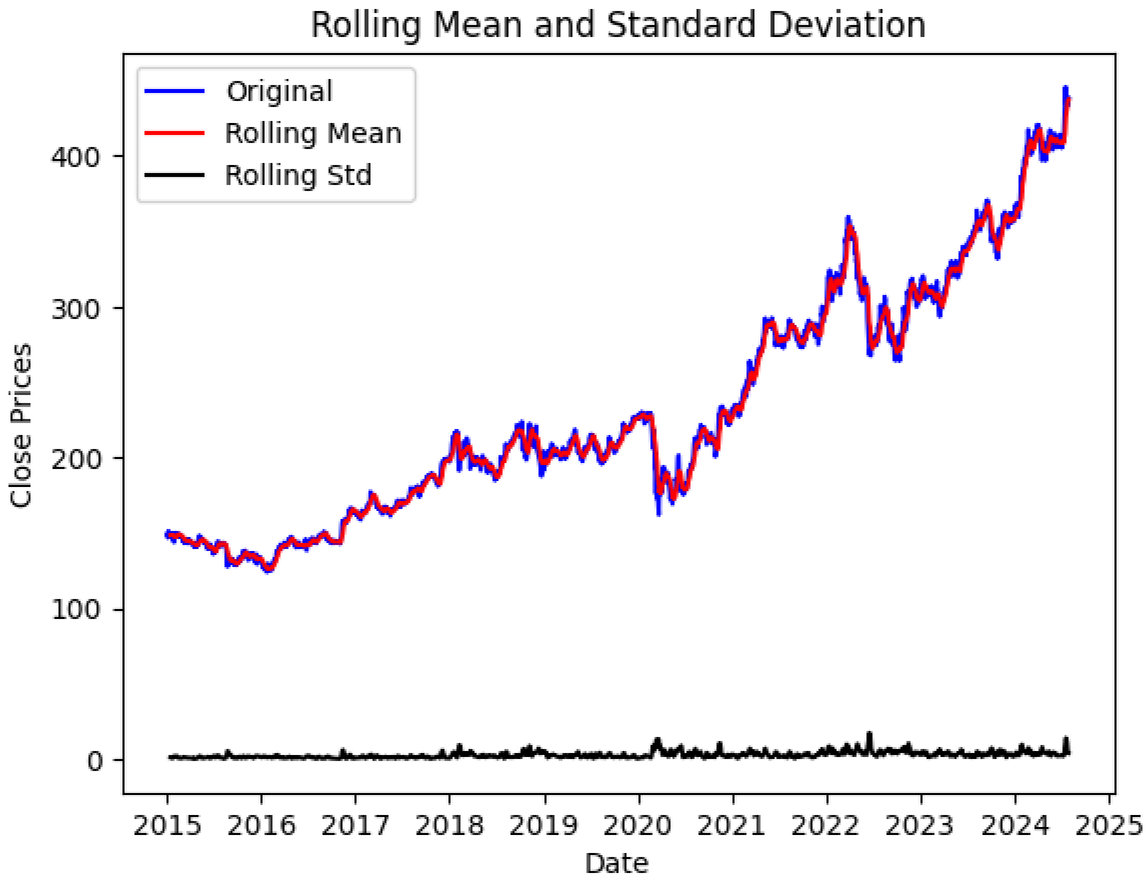
Stationarity Testing

Most time series forecasting models like ARIMA and SARIMA require the data to be stationary to make accurate predictions. This means that the data has a constant mean, variance, and autocorrelation over time. We perform a Dickey-Fuller Test to determine if our dataset is stationary.

In [80]:

```
#Determing rolling statistics
rolmean = df_close.rolling(10).mean()
rolstd = df_close.rolling(10).std()

#Plot rolling statistics
plt.plot(df_close, color='blue',label='Original')
plt.plot(rolmean, color='red', label='Rolling Mean')
plt.plot(rolstd, color='black', label = 'Rolling Std')
plt.xlabel('Date')
plt.ylabel('Close Prices')
plt.legend(loc='best')
plt.title('Rolling Mean and Standard Deviation')
plt.show(block=False)

# Perform the test and print results.
adft = adfuller(df_close,autolag='AIC')
```

```
print("Result:")
print("Dickey-Fuller Test Results:")
print(f"Test Statistic: {adft[0]}")
print(f"p-value: {adft[1]}")
```




```
Result:
Dickey-Fuller Test Results:
Test Statistic: 0.7636122089897024
p-value: 0.9910255061533995
```

Since the p-value > 0.05, this means the data is non-stationary. This means we will have to apply transformations like log scaling before fitting our model. The graph also shows us that both the rolling mean and STD are not stable, supporting our test.

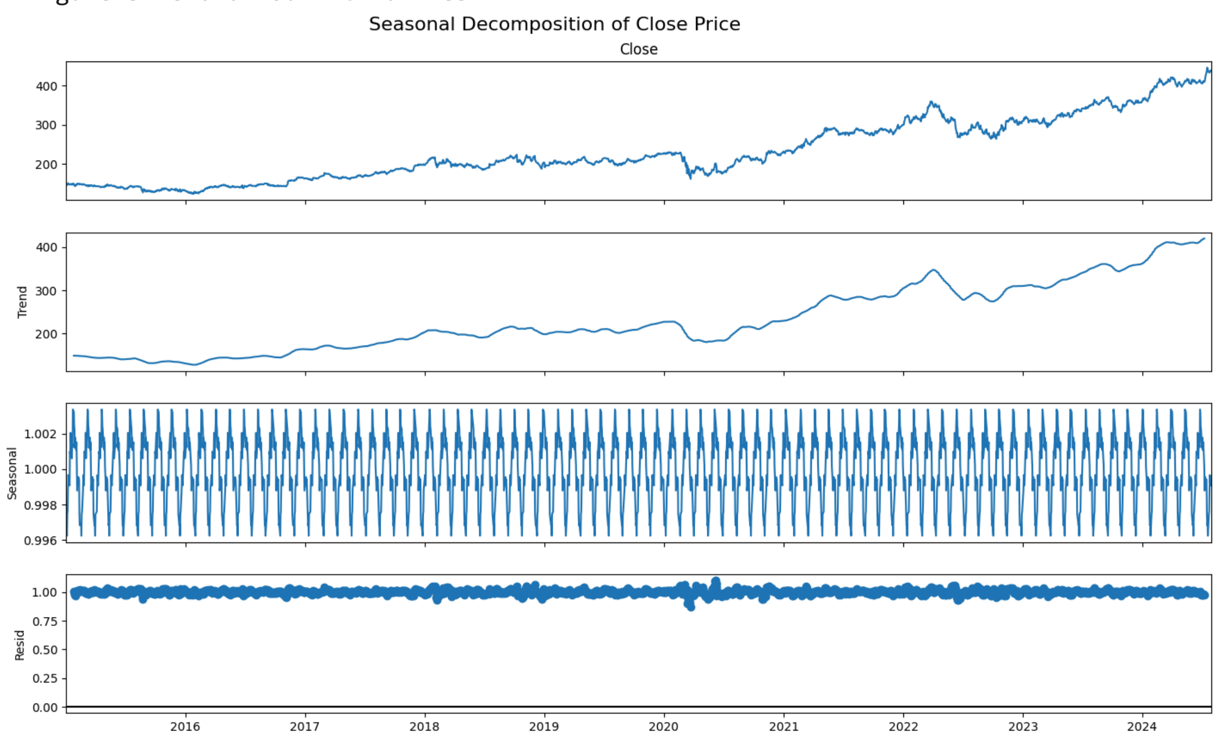
Seasionality Decomposition & Log Transformations

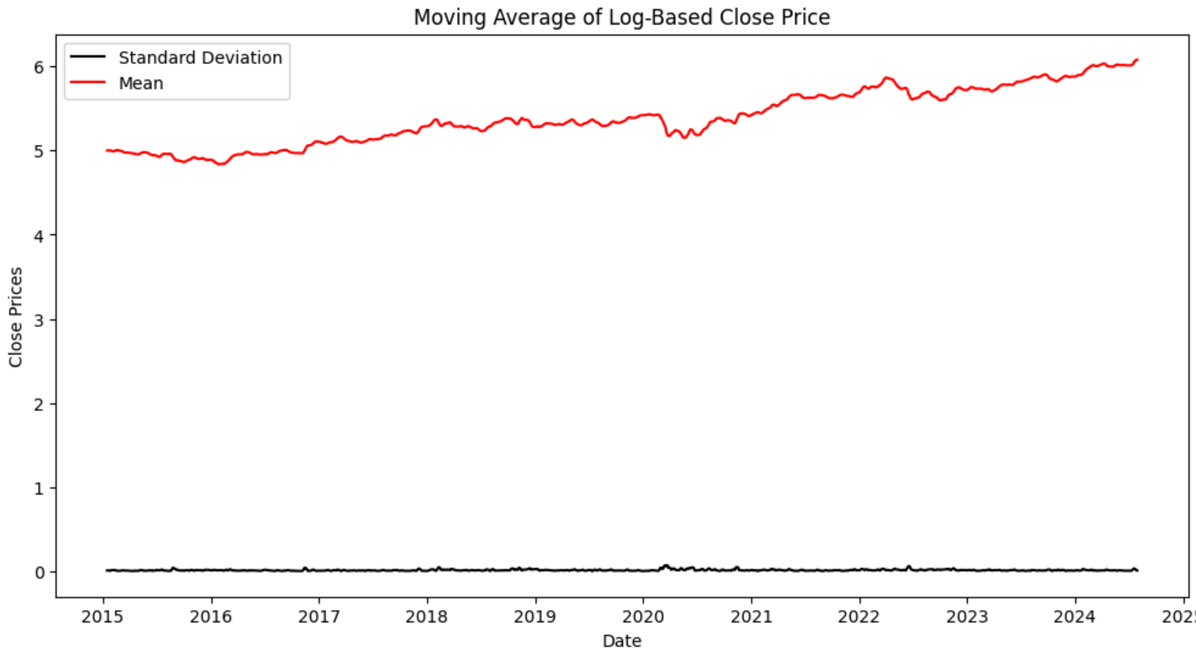
Since our data failed our test, we will have to process our data further. We will seperate the time series into trend, seasonality, and noise. This helped us discover the need for SARIMA later on as it handles this much better. However, for visualization this is still important.

To stablize our data and remove some of the variance, we use log transformation. This helps make the data more stationary. Then we plot to determine if it remains constant over time.

In [85]:
```
# Perform seasonal decomposition to seperate the time series into relevant parts (t
result = seasonal_decompose(df_close, model='multiplicative', period = 30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16, 9)
fig.suptitle("Seasonal Decomposition of Close Price", fontsize=16)

# Apply log transformations
df_log = np.log(df_close)
moving_avg = df_log.rolling(10).mean()
std_dev = df_log.rolling(10).std()

# Plot Log Transformation
plt.figure(figsize=(12,6))
plt.legend(loc='best')
plt.title('Moving Average of Log-Based Close Price')
plt.plot(std_dev, color ="black", label = "Standard Deviation")
plt.plot(moving_avg, color="red", label = "Mean")
plt.xlabel('Date')
plt.ylabel('Close Prices')
plt.legend()
plt.show()
```

```
C:\Users\Danyil\AppData\Local\Temp\ipykernel_38936\103077772.py:15: UserWarning: No
artists with labels found to put in legend.  Note that artists whose label start wit
h an underscore are ignored when legend() is called with no argument.
  plt.legend(loc='best')
<Figure size 640x480 with 0 Axes>
```





Split Data (Train-Test) & Auto ARIMA

Originally, wse manually calculated our ARIMA values. This left us with underwhelming results. So to improve on our previous model we prepare the dataset for auto ARIMA to get more accurate predictions and confirm the best model configuration.

We split the data by the 80/20 rule. This gives the model enough data to train on and leaves us with a good amount to test with.

In [86]:

```
# Split by the 80/20 rule
train, test = df_log[:int(len(df_log)*0.8)], df_log[int(len(df_log)*0.8):]

# Plot the data and show the test data
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Closing Prices')
plt.plot(df_log, 'green', label='Train data')
plt.plot(test, 'blue', label='Test data')
plt.title("Training vs Testing Log Close Price")
plt.legend()
```

Out[86]:

```
<matplotlib.legend.Legend at 0x2139d3ceb90>
```

In [38]:

```
# Train Auto ARIMA values
model_autoARIMA = auto_arima(train, start_p=0, start_q=0,
                      test='adf',
                      max_p=3, max_q=3,
                      m=1,
                      d=None,
                      seasonal=False,
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)

# Print out result
print(model_autoARIMA.summary())
model_autoARIMA.plot_diagnostics(figsize=(15,8))
plt.show()
```

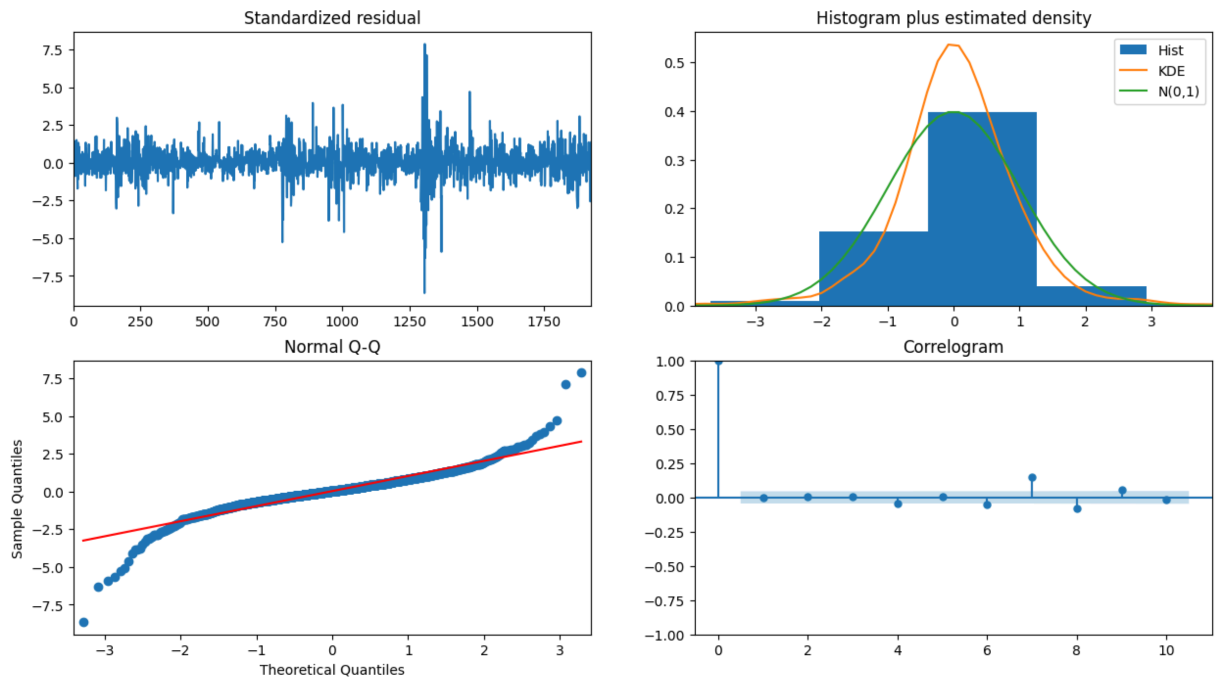
```
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```

```
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```

```
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-11328.005, Time=0.19 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-11362.955, Time=0.12 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-11358.441, Time=0.20 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-11328.473, Time=0.07 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-11367.278, Time=0.31 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-11365.724, Time=0.33 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-11364.962, Time=0.58 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-11364.536, Time=0.30 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-11363.547, Time=0.39 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=-11367.476, Time=0.15 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(1,1,0)(0,0,0)[0]             : AIC=-11362.935, Time=0.09 sec
 ARIMA(3,1,0)(0,0,0)[0]             : AIC=-11365.975, Time=0.06 sec
```

```
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(2,1,1)(0,0,0)[0]             : AIC=-11365.122, Time=0.16 sec
 ARIMA(1,1,1)(0,0,0)[0]             : AIC=-11364.586, Time=0.13 sec
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\ut
ils\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all
_finite' in 1.6 and will be removed in 1.8.
  warnings.warn(
 ARIMA(3,1,1)(0,0,0)[0]             : AIC=-11363.792, Time=0.20 sec

Best model:  ARIMA(2,1,0)(0,0,0)[0]
Total fit time: 3.306 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1926
Model:               SARIMAX(2, 1, 0)   Log Likelihood                5686.738
Date:                Thu, 24 Apr 2025   AIC                         -11367.476
Time:                        02:21:26   BIC                         -11350.788
Sample:                             0   HQIC                        -11361.337
                               - 1926
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.1291      0.010    -13.358      0.000      -0.148      -0.110
ar.L2          0.0583      0.011      5.516      0.000       0.038       0.079
sigma2         0.0002   2.21e-06     71.939      0.000       0.000       0.000
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):              8087.71
Prob(Q):                              0.94   Prob(JB):                         0.00
Heteroskedasticity (H):               2.80   Skew:                            -0.40
Prob(H) (two-sided):                  0.00   Kurtosis:                        13.01
===================================================================================

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



Train SARIMA Model & Predict

With our data explored, preprocessed, and the best configuration found we are ready to train our model. After testing both ARIMA and SARIMA, we noticed that the SARIMA performed much better. Which was one way that we improved our model.

In [39]:

```
# Train SARIMA model using optimal parameters from auto_arima
# To account for non-zero mean, we add a constant 'c' term
model = SARIMAX(train, order=model_autoARIMA.order, trend='c')

# Fit the model and show summary.
fitted = model.fit()
print(fitted.summary())

# Predict the length of the test values with 95% confidence intervals
forecast_result = fitted.get_forecast(len(test), alpha=0.05)

# Extract predicted mean, lower and upper bounds of the confidence interval, and st
fc = forecast_result.predicted_mean
conf = forecast_result.conf_int()
se = forecast_result.se_mean
```

```
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodel
s\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it ha
s no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodel
s\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provided, but it ha
s no associated frequency information and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

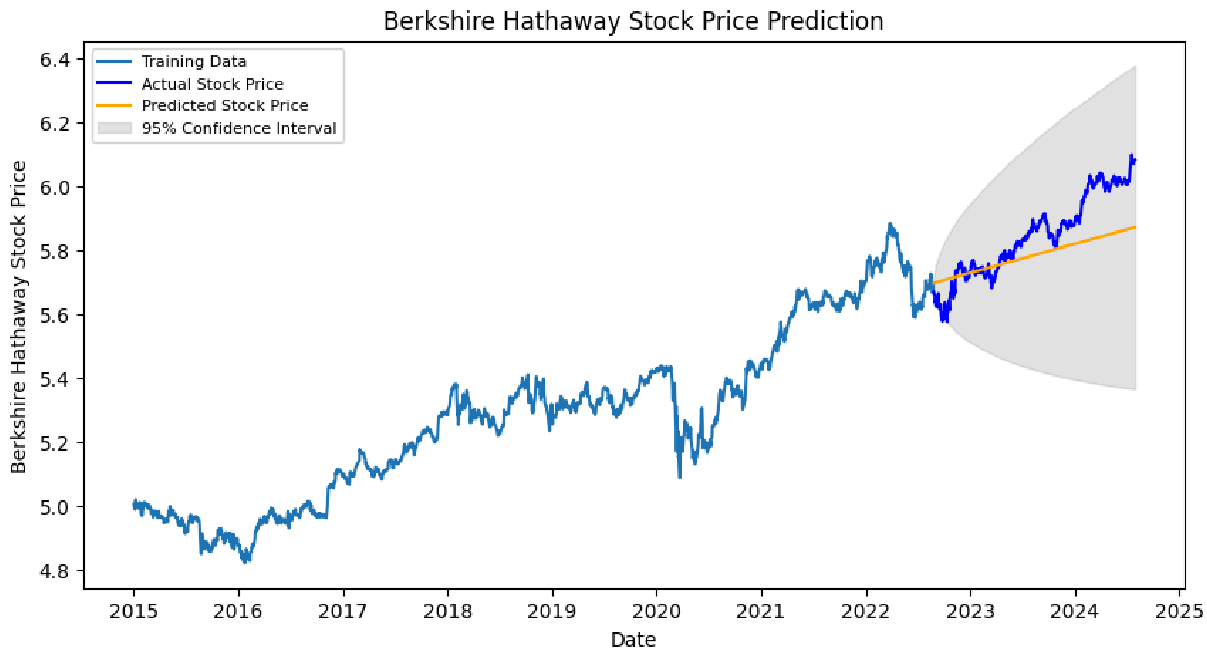
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  Close   No. Observations:                 1926
Model:               SARIMAX(2, 1, 0)   Log Likelihood                5687.639
Date:                Thu, 24 Apr 2025   AIC                         -11367.278
Time:                        02:21:27   BIC                         -11345.028
Sample:                             0   HQIC                        -11359.093
                               - 1926
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0004      0.000      1.269      0.204      -0.000       0.001
ar.L1         -0.1301      0.010    -12.954      0.000      -0.150      -0.110
ar.L2          0.0573      0.011      5.250      0.000       0.036       0.079
sigma2         0.0002   2.24e-06     70.761      0.000       0.000       0.000
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):              8070.03
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               2.80   Skew:                            -0.40
Prob(H) (two-sided):                  0.00   Kurtosis:                        13.00
===================================================================================

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

```
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodel
s\tsa\base\tsa_model.py:837: ValueWarning: No supported index is available. Predicti
on results will be given with an integer index beginning at `start`.
  return get_prediction_index(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodel
s\tsa\base\tsa_model.py:837: FutureWarning: No supported index is available. In the
next version, calling this method in a model without a supported index will result i
n an exception.
  return get_prediction_index(
c:\Users\Danyil\AppData\Local\Programs\Python\Python311\Lib\site-packages\statsmodel
s\tsa\statespace\representation.py:374: FutureWarning: Unknown keyword arguments: di
ct_keys(['alpha']).Passing unknown keyword arguments will raise a TypeError beginnin
g in version 0.15.
  warnings.warn(msg, FutureWarning)
```

In [88]:

```
# Plot our predictions

plt.figure(figsize=(10,5), dpi=100)

# Plot training data
plt.plot(train.index, train.values, label='Training Data')

# Plot test data
plt.plot(test.index, test.values, color='blue', label='Actual Stock Price')

# Plot forecast values
plt.plot(test.index, fc, color='orange', label='Predicted Stock Price')

# Plot confidence intervals
plt.fill_between(test.index, conf.iloc[:, 0], conf.iloc[:, 1], color='k', alpha=0.1
```

```
plt.title('Berkshire Hathaway Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Berkshire Hathaway Stock Price')
plt.legend(loc='upper left', fontsize=8)
plt.show()
```



Evaluate Model Performance

Now for the moment of truth, we see how well our model performed using MSE, MAE, and RMSE

In [41]:
```
mse = mean_squared_error(test, fc)
mae = mean_absolute_error(test, fc)
rmse = math.sqrt(mean_squared_error(test, fc))
print('Evaluating: SARIMA')
print('MSE: '+ str(mse))
print('MAE: '+ str(mae))
print('RMSE: ' +str(rmse))
```

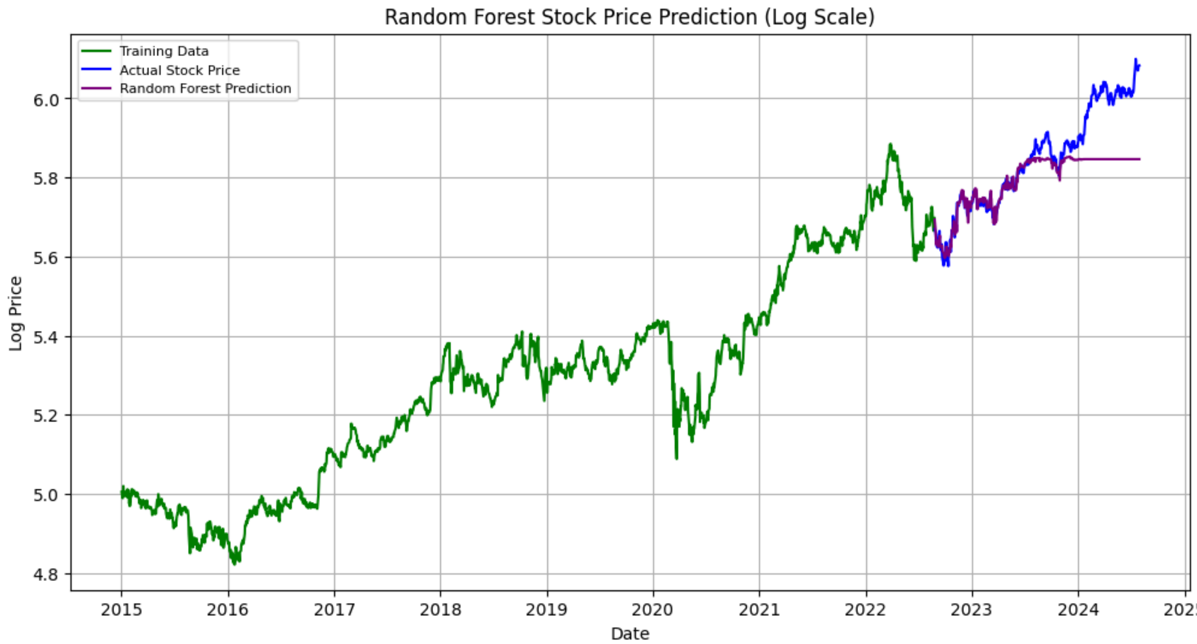
```
Evaluating: SARIMA
MSE: 0.00999018901692493
MAE: 0.08015484728919989
RMSE: 0.09995093304679517
```

Train Random Forest

In addition to SARIMA, we train a Random Forest model on the same dataset. This model will help capture more complex patterns that statistical models like SARIMA might miss, potentially improving our prediction accuracy when combined in an ensemble.

In [46]:
```
def create_features(data, window_size=30):
    X, y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i+window_size])
        y.append(data[i+window_size])
    return np.array(X), np.array(y)

# Create features for Random Forest
window_size = 30
X_train, y_train = create_features(train.values, window_size)
X_test = []
for i in range(len(train) - window_size, len(train) - window_size + len(test)):
    if i + window_size <= len(df_log):
        X_test.append(df_log.values[i:i+window_size])
X_test = np.array(X_test)

# Train Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions with Random Forest
rf_predictions = rf_model.predict(X_test)
```

In [53]:
```
# Plot Random Forest Model
plt.figure(figsize=(12, 6))
plt.plot(train.index, train, label='Training Data', color='green')
plt.plot(test.index, test, label='Actual Stock Price', color='blue')
plt.plot(test.index[:len(rf_predictions)], rf_predictions[:len(test.index)], label=

plt.title('Random Forest Stock Price Prediction (Log Scale)')
plt.xlabel('Date')
plt.ylabel('Log Price')
plt.legend(loc='upper left', fontsize=8)
plt.grid(True)
plt.show()
```



Evaluate Random Forest

In [59]:
```
rf_mse = mean_squared_error(test[:len(rf_predictions)], rf_predictions)
rf_mae = mean_absolute_error(test[:len(rf_predictions)], rf_predictions)
rf_rmse = math.sqrt(rf_mse)

print("Results: Random Forest")
print('MSE: ' + str(rf_mse))
print('MAE: ' + str(rf_mae))
print('RMSE: ' + str(rf_rmse))
```

```
Results: Random Forest
MSE: 0.008127500004852637
MAE: 0.057756966772724475
RMSE: 0.09015264835185174
```

Results

As seen in the graph, the random forest model captures the up and down nature of the stock price much more closely compared to SARIMA, but struggles towards the end. Our goal is to combine both of these models.

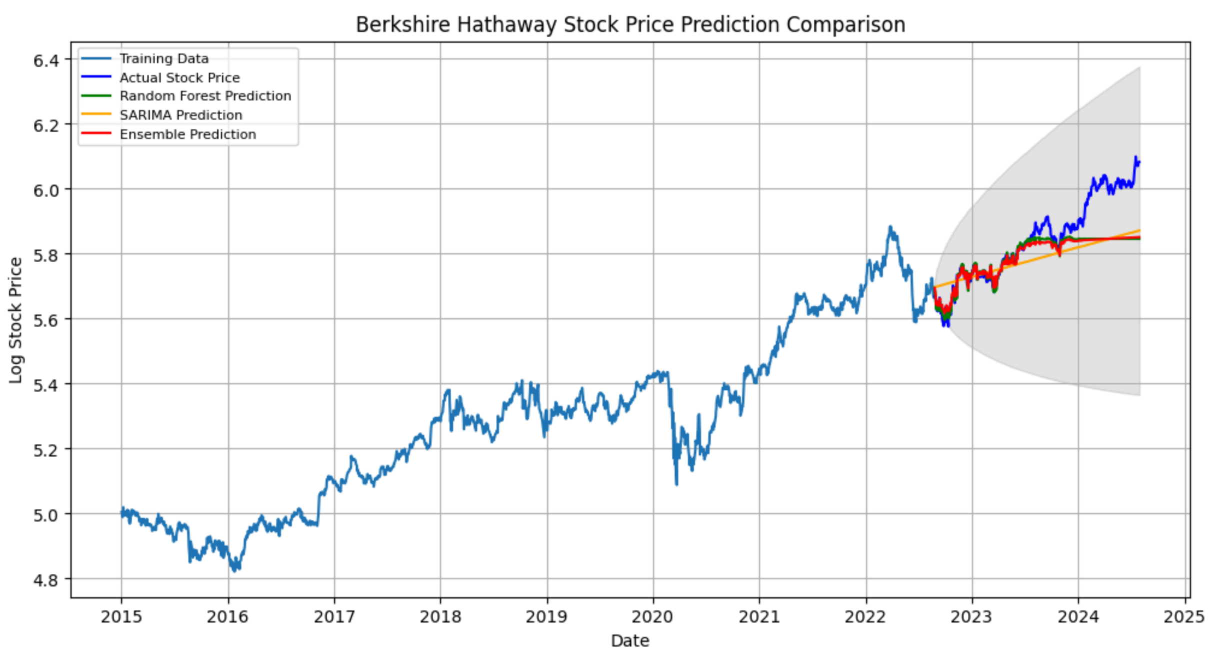
Ensemble Time!

Now it's time to combine our two models to create a more accurate one. The idea was to capture the strengths of both models. SARIMA captures linear and seasonal trends, while Random Forest captures non-linear patterns.

In [77]:
```
# Use weights to tune the model
sarima_weight = 0.2
rf_weight = 0.8

ensemble_predictions = (sarima_weight * fc.values) + (rf_weight * rf_predictions)

# Evaluate Performance
ensemble_mse = mean_squared_error(test[:len(ensemble_predictions)], ensemble_predic
ensemble_mae = mean_absolute_error(test[:len(ensemble_predictions)], ensemble_predi
ensemble_rmse = math.sqrt(ensemble_mse)

print("Results: Ensemble")
print('MSE: ' + str(ensemble_mse))
print('MAE: ' + str(ensemble_mae))
print('RMSE: ' + str(ensemble_rmse))
```

```
Results: Ensemble
MSE: 0.008249708751092438
MAE: 0.06039318276112974
RMSE: 0.09082790733630516
```

In [87]:
```
# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(train.index, train.values, label='Training Data')
plt.plot(test.index, test.values, color='blue', label='Actual Stock Price')
plt.plot(test.index[:len(rf_predictions)], rf_predictions, color='green', label='Ra
plt.plot(test.index, fc, color='orange', label='SARIMA Prediction')
plt.plot(test.index[:len(ensemble_predictions)], ensemble_predictions, color='red',
plt.fill_between(test.index, conf.iloc[:, 0], conf.iloc[:, 1], color='k', alpha=0.1
plt.title('Berkshire Hathaway Stock Price Prediction Comparison')
plt.xlabel('Date')
plt.ylabel('Log Stock Price')
plt.legend(loc='upper left', fontsize=8)
plt.grid(True)
plt.show()
```



Bar Graph Comparing Models

In [79]:
```
# Create a bar chart to compare MSE values
models = ['SARIMA', 'Random Forest', 'Ensemble']
mse_values = [rmse, rf_mse, ensemble_mse]
rmse_values = [rmse, rf_rmse, ensemble_rmse]

plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
plt.bar(models, mse_values, color=['orange', 'green', 'red'])
plt.title('MSE Comparison')
plt.ylabel('Mean Squared Error')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.subplot(1, 2, 2)
plt.bar(models, rmse_values, color=['orange', 'green', 'red'])
plt.title('RMSE Comparison')
plt.ylabel('Root Mean Squared Error')
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()
plt.show()
```