In [1]:
```
import numpy as np
import pandas as pd
import yfinance as yf #use to fetch stock data
import matplotlib.pyplot as plt
```

In [2]:
```
portfolio_commodities = ["GC=F","SI=F","CL=F"]
```

In [3]:
```
data = yf.download(portfolio_commodities, start='2023-12-01', end='2024-12-01')['Adj Close']
data
```

```
[*********************100%%**********************]  3 of 3 completed
```

Out[3]:

Ticker	CL=F	GC=F	SI=F
Date			
2023-12-01	74.070000	2071.000000	25.499001
2023-12-04	73.040001	2024.099976	24.555000
2023-12-05	72.320000	2018.500000	24.201000
2023-12-06	69.379997	2030.500000	23.889000
2023-12-07	69.339996	2029.900024	23.732000
...	...	...	...
2024-11-22	71.239998	2709.899902	31.309000
2024-11-25	68.940002	2616.800049	30.209999
2024-11-26	68.769997	2620.300049	30.388000
2024-11-27	68.720001	2639.899902	30.111000
2024-11-29	68.000000	2657.000000	30.684999

251 rows × 3 columns

In [6]:
```
commodities_returns= data.pct_change().dropna()
commodities_returns.head()
```

Out[6]:

Ticker	CL=F	GC=F	SI=F
Date			
2023-12-04	-0.013906	-0.022646	-0.037021
2023-12-05	-0.009858	-0.002767	-0.014417
2023-12-06	-0.040653	0.005945	-0.012892
2023-12-07	-0.000577	-0.000295	-0.006572
2023-12-08	0.027257	-0.015567	-0.032066

In [7]:
```
weights_commodities = np.array([0.34, 0.33, 0.33])
```

In [8]:
```
# Calculate Expected Daily Return
portfolio_return_commodities = commodities_returns.dot(weights_commodities)
expected_returns_commodities = portfolio_return_commodities.mean()
```

In [9]:
```
# Calculate covariance matrix of returns
commodities_cov_matrix = commodities_returns.cov()

# Calculate portfolio variance and volatility
portfolio_variance = np.dot(weights_commodities.T, np.dot(commodities_cov_matrix, weights_commodities))
portfolio_volatility = np.sqrt(portfolio_variance)

# Print results
print(f"Expected Daily Return commodities: {expected_returns_commodities:.4f}")
print(f"Portfolio Variance commodities: {portfolio_variance:.6f}")
print(f"Portfolio Volatility commodities(Risk): {portfolio_volatility:.4f}")
```

```
Expected Daily Return commodities: 0.0006
Portfolio Variance commodities: 0.000156
Portfolio Volatility commodities(Risk): 0.0125
```

In [10]:
```
portfolio_Shares= ["RELIANCE.NS","TCS.NS","HDFCBANK.NS"]
```

In [11]:
```
data = yf.download(portfolio_Shares, start='2023-12-01', end='2024-12-01')['Adj Close']
data
```

```
[*********************100%%**********************]  3 of 3 completed
```

Out[11]:

Ticker	HDFCBANK.NS	RELIANCE.NS	TCS.NS
Date			
2023-12-01	1534.446533	1189.051270	3461.765625
2023-12-04	1587.718994	1201.913696	3462.554443
2023-12-05	1601.826294	1210.629272	3481.432373
2023-12-06	1605.871216	1222.225342	3552.902588
2023-12-07	1608.485352	1220.213989	3563.548828
...	...	...	...
2024-11-25	1785.599976	1287.000000	4315.100098
2024-11-26	1785.550049	1295.699951	4352.700195
2024-11-27	1812.300049	1293.199951	4332.549805
2024-11-28	1793.150024	1270.800049	4244.899902
2024-11-29	1796.050049	1292.199951	4270.850098

245 rows × 3 columns

In [12]:
```
share_returns= data.pct_change().dropna()
share_returns.head()
```

Out[12]:

Ticker	HDFCBANK.NS	RELIANCE.NS	TCS.NS
Date			
2023-12-04	0.034718	0.010817	0.000228
2023-12-05	0.008885	0.007251	0.005452
2023-12-06	0.002525	0.009579	0.020529
2023-12-07	0.001628	-0.001646	0.002996
2023-12-08	0.013953	-0.000529	0.003264

In [13]:
```
weights_shares = np.array([0.34, 0.33, 0.33])
```

In [14]:
```
# Calculate Expected Daily Return
portfolio_return_shares = share_returns.dot(weights_shares)
expected_returns_shares = portfolio_return_shares.mean()
```

In [15]:
```
# Calculate covariance matrix of returns
shares_cov_matrix = share_returns.cov()

# Calculate portfolio variance and volatility
portfolio_variance = np.dot(weights_shares.T, np.dot(shares_cov_matrix, weights_shares))
portfolio_volatility = np.sqrt(portfolio_variance)

# Print results
print(f"Expected Daily Return shares: {expected_returns_shares:.4f}")
print(f"Portfolio Variance shares: {portfolio_variance:.6f}")
print(f"Portfolio Volatility shares (Risk): {portfolio_volatility:.4f}")
```

```
Expected Daily Return shares: 0.0007
Portfolio Variance shares: 0.000093
Portfolio Volatility shares (Risk): 0.0096
```

In [16]:
```
portfolio_Crypto= ["BTC-USD","ETH-USD","USDT-USD"]
```

In [17]:
```
data = yf.download(portfolio_Crypto, start='2023-12-01', end='2024-12-01')['Adj Close']
data
```

```
[**********100%%**********]  3 of 3 completed
```

Out[17]:

Ticker	BTC-USD	ETH-USD	USDT-USD
Date			
2023-12-01	38688.750000	2087.139893	1.000185
2023-12-02	39476.332031	2165.704102	1.000364
2023-12-03	39978.390625	2193.691650	1.000309
2023-12-04	41980.097656	2243.215820	0.999913
2023-12-05	44080.648438	2293.841797	1.000546
...	...	...	...
2024-11-26	91985.320312	3326.517334	0.999654
2024-11-27	95962.531250	3657.249268	1.000994
2024-11-28	95652.468750	3579.811523	1.000154
2024-11-29	97461.523438	3593.494385	1.000366
2024-11-30	96449.054688	3705.705322	1.000630

366 rows × 3 columns

In [18]:
```
crypto_returns= data.pct_change().dropna()
crypto_returns.head()
```

Out[18]:

Ticker	BTC-USD	ETH-USD	USDT-USD
Date			
2023-12-02	0.020357	0.037642	0.000179
2023-12-03	0.012718	0.012923	-0.000055
2023-12-04	0.050070	0.022576	-0.000396
2023-12-05	0.050037	0.022568	0.000633
2023-12-06	-0.007582	-0.027108	-0.000514

In [19]:
```
weights_crypto = np.array([0.34, 0.33, 0.33])
```

In [20]:
```
# Calculate Expected Daily Return
portfolio_return_crypto = crypto_returns.dot(weights_crypto)
expected_returns_crypto = portfolio_return_crypto.mean()
```

In [21]:
```
# Calculate covariance matrix of returns
crypto_cov_matrix = crypto_returns.cov()

# Calculate portfolio variance and volatility
portfolio_variance = np.dot(weights_crypto.T, np.dot(crypto_cov_matrix , weights_crypto))
portfolio_volatility = np.sqrt(portfolio_variance)

# Print results
print(f"Expected Daily Return crytp: {expected_returns_crypto:.4f}")
print(f"Portfolio Variance crypto: {portfolio_variance:.6f}")
print(f"Portfolio Volatility crypto (Risk): {portfolio_volatility:.4f}")
```

```
Expected Daily Return crytp: 0.0017
Portfolio Variance crypto: 0.000382
Portfolio Volatility crypto (Risk): 0.0196
```

In [22]:
```
VaR_95 = np.percentile(commodities_returns, 5)
print(f"Value at Risk commodities(5%): {VaR_95:.4f}")
VaR_95 = np.percentile(share_returns, 5)
print(f"Value at Risk shares(5%): {VaR_95:.4f}")
VaR_95 = np.percentile(crypto_returns, 5)
print(f"Value at Risk crypto(5%): {VaR_95:.4f}")
```

```
Value at Risk commodities(5%): -0.0312
Value at Risk shares(5%): -0.0190
Value at Risk crypto(5%): -0.0390
```

In [23]:
```
# Calculate cumulative returns
cumulative_commodities = (1 + portfolio_return_commodities).cumprod() - 1
cumulative_crypto = (1 + portfolio_return_crypto).cumprod() - 1
cumulative_equities = (1 + portfolio_return_shares).cumprod() - 1
```

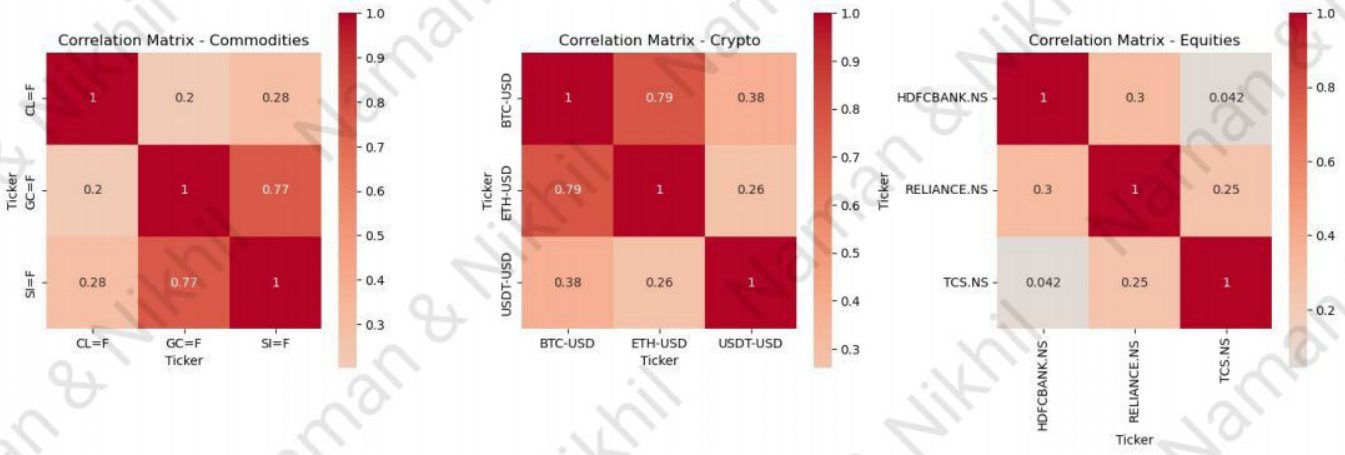
In [24]:
```
# Plotting the cumulative returns
plt.figure(figsize=(12, 6))
plt.plot(cumulative_commodities, label='Commodities Portfolio', linewidth=2)
plt.plot(cumulative_crypto, label='Crypto Portfolio', linewidth=2)
plt.plot(cumulative_equities, label='Equities Portfolio', linewidth=2)

# Add title and labels
plt.title('Portfolio Cumulative Returns Comparison', fontsize=16)
plt.xlabel('Date', fontsize=14)
```

```
plt.ylabel('Cumulative Return', fontsize=14)
plt.legend(loc='upper left')
plt.grid(True)

# Show the graph
plt.show()
```

In [25]:
```
# Calculate expected returns
expected_return_commodities = portfolio_return_commodities.mean()
expected_return_crypto = portfolio_return_crypto.mean()
expected_return_shares = portfolio_return_shares.mean()

# Calculate variance and standard deviation
variance_commodities = portfolio_return_commodities.var()
variance_crypto = portfolio_return_crypto.var()
variance_shares = portfolio_return_shares.var()

std_dev_commodities = np.sqrt(variance_commodities)
std_dev_crypto = np.sqrt(variance_crypto)
std_dev_shares = np.sqrt(variance_shares)

# Print results
print(f"Expected Return Commodities: {expected_return_commodities:.4f}, Volatility: {std_dev_commodities:.4f}")
print(f"Expected Return Crypto: {expected_return_crypto:.4f}, Volatility: {std_dev_crypto:.4f}")
print(f"Expected Return shares: {expected_return_shares:.4f}, Volatility: {std_dev_shares:.4f}")
```

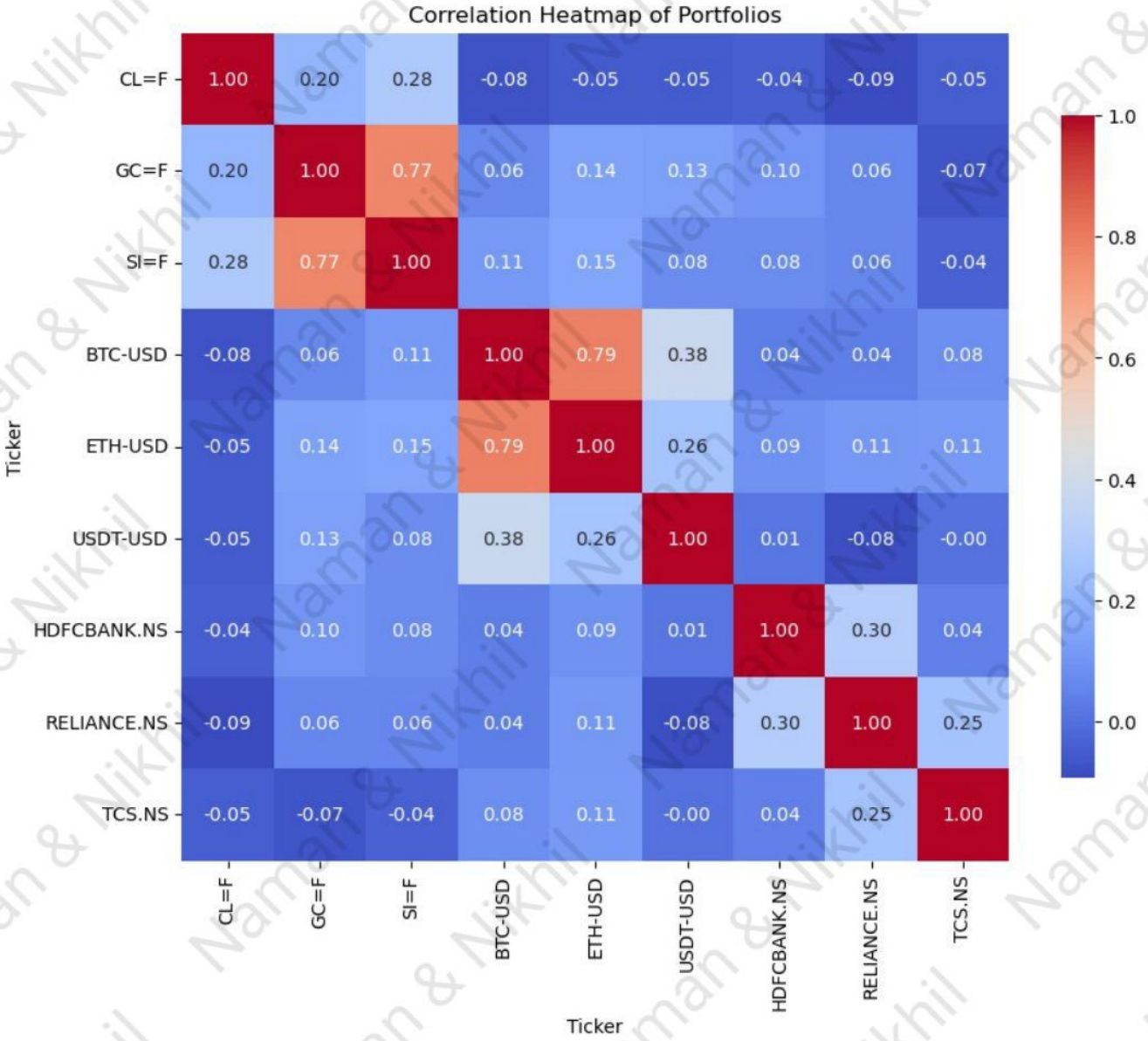
```
Expected Return Commodities: 0.0006, Volatility: 0.0125
Expected Return Crypto: 0.0017, Volatility: 0.0196
Expected Return shares: 0.0007, Volatility: 0.0096
```

In [26]:
```
import seaborn as sns
# Calculate correlation matrices
correlation_commodities = commodities_returns.corr()
correlation_crypto = crypto_returns.corr()
correlation_share = share_returns.corr()

# Plotting the correlation matrices as heatmaps
plt.figure(figsize=(15, 5))

# Heatmap for Commodities
plt.subplot(1, 3, 1)
sns.heatmap(correlation_commodities, annot=True, cmap='coolwarm', center=0, cbar=True, square=True)
plt.title('Correlation Matrix - Commodities')

# Heatmap for Crypto
plt.subplot(1, 3, 2)
sns.heatmap(correlation_crypto, annot=True, cmap='coolwarm', center=0, cbar=True, square=True)
plt.title('Correlation Matrix - Crypto')

# Heatmap for Equities
plt.subplot(1, 3, 3)
sns.heatmap(correlation_share, annot=True, cmap='coolwarm', center=0, cbar=True, square=True)
plt.title('Correlation Matrix - Equities')

# Adjust layout
plt.tight_layout()
plt.show()
```



In [27]:

```
# Assuming you have the returns data for each portfolio
# commodities_returns, crypto_returns, and share_returns are your DataFrames

# Example DataFrames (replace these with your actual returns DataFrames)
# commodities_returns = pd.DataFrame(...)  # Your commodities returns DataFrame
# crypto_returns = pd.DataFrame(...)       # Your crypto returns DataFrame
# share_returns = pd.DataFrame(...)        # Your shares returns DataFrame

# Concatenate the returns DataFrames
combined_returns = pd.concat([commodities_returns, crypto_returns, share_returns], axis=1)

# Calculate the correlation matrix
correlation_matrix = combined_returns.corr()

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Create a heatmap
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True, cbar_kws={"shrink": .8})

# Set the title
plt.title('Correlation Heatmap of Portfolios')

# Show the plot
plt.show()
```



In [34]:
```
# Thank You
# Naman Sharma
```