In [58]:

```
import pandas as pd
import numpy as np

data = pd.read_csv('Downloads/archive (4)/South_Asian_dataset.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 192 entries, 0 to 191
Data columns (total 33 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   Country
192 non-null    object
 1   Year
192 non-null    int64
 2   GDP (current US$)
190 non-null    float64
 3   GDP growth (annual %)
189 non-null    float64
 4   GDP per capita (current US$)
190 non-null    float64
 5   Unemployment, total (% of total labor force) (modeled ILO estimate)
192 non-null    float64
 6   Inflation, consumer prices (annual %)
183 non-null    float64
 7   Foreign direct investment, net inflows (% of GDP)
187 non-null    float64
 8   Trade (% of GDP)
141 non-null    float64
 9   Gini index
42 non-null     float64
 10  Population, total
192 non-null    int64
 11  Population growth (annual %)
192 non-null    float64
 12  Poverty headcount ratio at $2.15 a day (2017 PPP) (% of population)
192 non-null    object
 13  Life expectancy at birth, total (years)
192 non-null    object
 14  Mortality rate, infant (per 1,000 live births)
192 non-null    object
 15  Literacy rate, adult total (% of people ages 15 and above)
192 non-null    object
 16  School enrollment, primary (% gross)
192 non-null    object
 17  Urban population (% of total population)
192 non-null    float64
 18  Access to electricity (% of population)
192 non-null    object
 19  People using at least basic drinking water services (% of population)
192 non-null    object
 20  People using at least basic sanitation services (% of population)
```

```
192 non-null    object
 21  Carbon dioxide (CO2) emissions excluding LULUCF per capita (t CO2e/ca
pita)  192 non-null    object
 22  PM2.5 air pollution, mean annual exposure (micrograms per cubic meter
)       192 non-null    object
 23  Renewable energy consumption (% of total final energy consumption)
192 non-null    object
 24  Forest area (% of land area)
192 non-null    object
 25  Control of Corruption: Percentile Rank
192 non-null    object
 26  Political Stability and Absence of Violence/Terrorism: Estimate
192 non-null    object
 27  Regulatory Quality: Estimate
192 non-null    object
 28  Rule of Law: Estimate
192 non-null    object
 29  Voice and Accountability: Estimate
192 non-null    object
 30  Individuals using the Internet (% of population)
192 non-null    object
 31  Research and development expenditure (% of GDP)
192 non-null    object
 32  High-technology exports (% of manufactured exports)
192 non-null    object
dtypes: float64(10), int64(2), object(21)
memory usage: 49.6+ KB
```

In [59]:

```
data.head()
```

Out[59]:

	Country	Year	GDP (current US$)	GDP growth (annual %)	GDP per capita (current US$)	Unemployment, total (% of total labor force) (modeled ILO estimate)	Infla consu p (ar
0	Afghanistan	2000	3.521418e+09	NaN	180.188369	7.955	
1	Afghanistan	2001	2.813572e+09	-9.431974	142.903364	7.958	
2	Afghanistan	2002	3.825701e+09	28.600001	182.174038	7.939	
3	Afghanistan	2003	4.520947e+09	8.832278	199.643226	7.922	
4	Afghanistan	2004	5.224897e+09	1.414118	221.830531	7.914	

5 rows × 33 columns

In [60]:

```
data.replace("..", np.nan, inplace=True)
# Converting columns to numeric and handling exceptions
for column in data.columns:
    try:
        data[column] = pd.to_numeric(data[column])
```

```
    except ValueError:
        pass # If conversion fails, the column is likely non-numeric; co

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 192 entries, 0 to 191
Data columns (total 33 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   Country
192 non-null    object
 1   Year
192 non-null    int64
 2   GDP (current US$)
190 non-null    float64
 3   GDP growth (annual %)
189 non-null    float64
 4   GDP per capita (current US$)
190 non-null    float64
 5   Unemployment, total (% of total labor force) (modeled ILO estimate)
192 non-null    float64
 6   Inflation, consumer prices (annual %)
183 non-null    float64
 7   Foreign direct investment, net inflows (% of GDP)
187 non-null    float64
 8   Trade (% of GDP)
141 non-null    float64
 9   Gini index
42 non-null     float64
 10  Population, total
192 non-null    int64
 11  Population growth (annual %)
192 non-null    float64
 12  Poverty headcount ratio at $2.15 a day (2017 PPP) (% of population)
42 non-null     float64
 13  Life expectancy at birth, total (years)
184 non-null    float64
 14  Mortality rate, infant (per 1,000 live births)
184 non-null    float64
 15  Literacy rate, adult total (% of people ages 15 and above)
61 non-null     float64
 16  School enrollment, primary (% gross)
166 non-null    float64
 17  Urban population (% of total population)
192 non-null    float64
 18  Access to electricity (% of population)
184 non-null    float64
 19  People using at least basic drinking water services (% of population)
184 non-null    float64
 20  People using at least basic sanitation services (% of population)
184 non-null    float64
 21  Carbon dioxide (CO2) emissions excluding LULUCF per capita (t CO2e/ca
pita)  184 non-null    float64
```

```
 22  PM2.5 air pollution, mean annual exposure (micrograms per cubic meter
)      168 non-null    float64
 23  Renewable energy consumption (% of total final energy consumption)
179 non-null    float64
 24  Forest area (% of land area)
176 non-null    float64
 25  Control of Corruption: Percentile Rank
176 non-null    float64
 26  Political Stability and Absence of Violence/Terrorism: Estimate
176 non-null    float64
 27  Regulatory Quality: Estimate
176 non-null    float64
 28  Rule of Law: Estimate
176 non-null    float64
 29  Voice and Accountability: Estimate
176 non-null    float64
 30  Individuals using the Internet (% of population)
173 non-null    float64
 31  Research and development expenditure (% of GDP)
48 non-null     float64
 32  High-technology exports (% of manufactured exports)
76 non-null     float64
dtypes: float64(30), int64(2), object(1)
memory usage: 49.6+ KB
```

In [61]:

```python
# Dropping columns because too many null values
data = data.drop(['Research and development expenditure (% of GDP)', 'Hi
data.head()
```

Out[61]:

	Country	Year	GDP (current US$)	GDP growth (annual %)	GDP per capita (current US$)	Unemployment, total (% of total labor force) (modeled ILO estimate)	Infla consu p (ar
0	Afghanistan	2000	3.521418e+09	NaN	180.188369	7.955	
1	Afghanistan	2001	2.813572e+09	-9.431974	142.903364	7.958	
2	Afghanistan	2002	3.825701e+09	28.600001	182.174038	7.939	
3	Afghanistan	2003	4.520947e+09	8.832278	199.643226	7.922	
4	Afghanistan	2004	5.224897e+09	1.414118	221.830531	7.914	

5 rows × 31 columns

In [62]:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

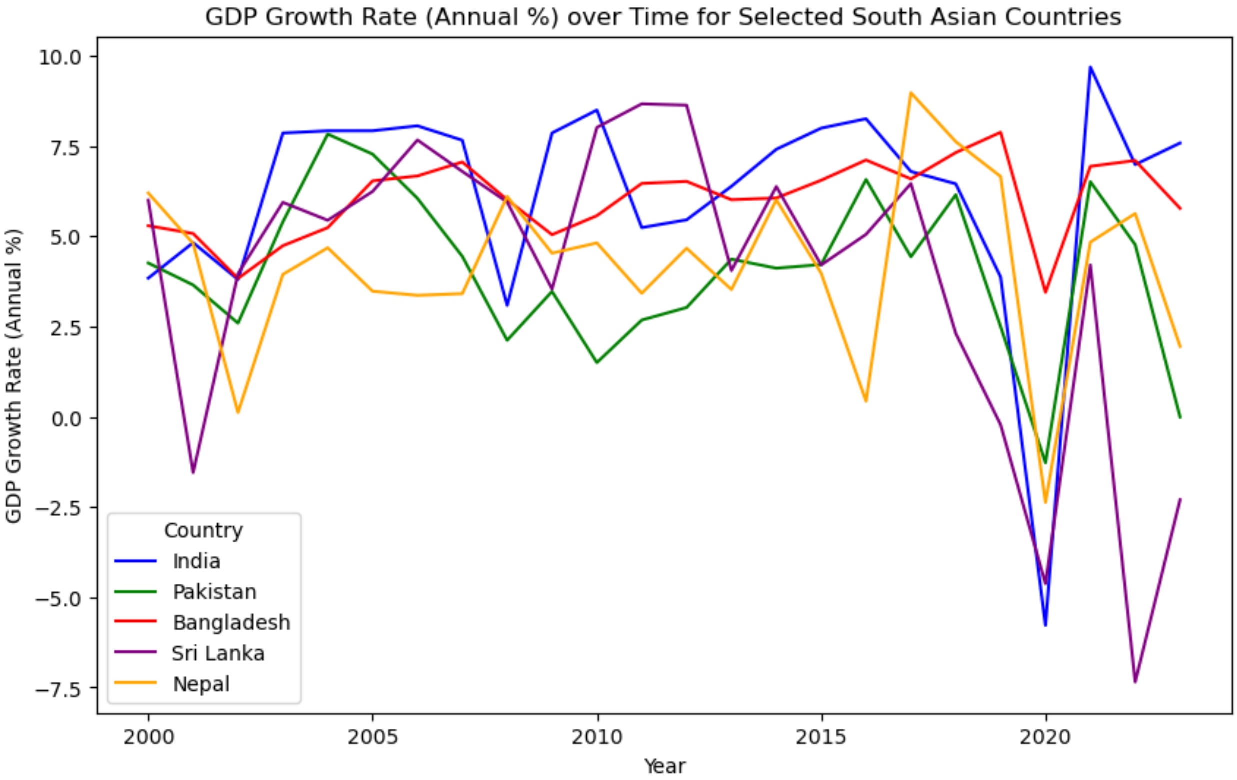
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

In [63]:

```
# Define the list of selected countries
selected_countries = ["India", "Pakistan", "Bangladesh", "Sri Lanka", "Ne

# Define color mapping for each country
country_colors = {
    "India": "blue",
    "Pakistan": "green",
    "Bangladesh": "red",
    "Sri Lanka": "purple",
    "Nepal": "orange"
}

# Select relevant columns for economic growth analysis
economic_growth_cols = [
    'Country', 'Year', 'GDP (current US$)', 'GDP growth (annual %)',
    'GDP per capita (current US$)',
    'Unemployment, total (% of total labor force) (modeled ILO estimate)
    'Inflation, consumer prices (annual %)'
]
economic_growth_data = data[economic_growth_cols]

# Filter the dataset for selected countries
economic_growth_filtered = economic_growth_data[economic_growth_data["Cou
```

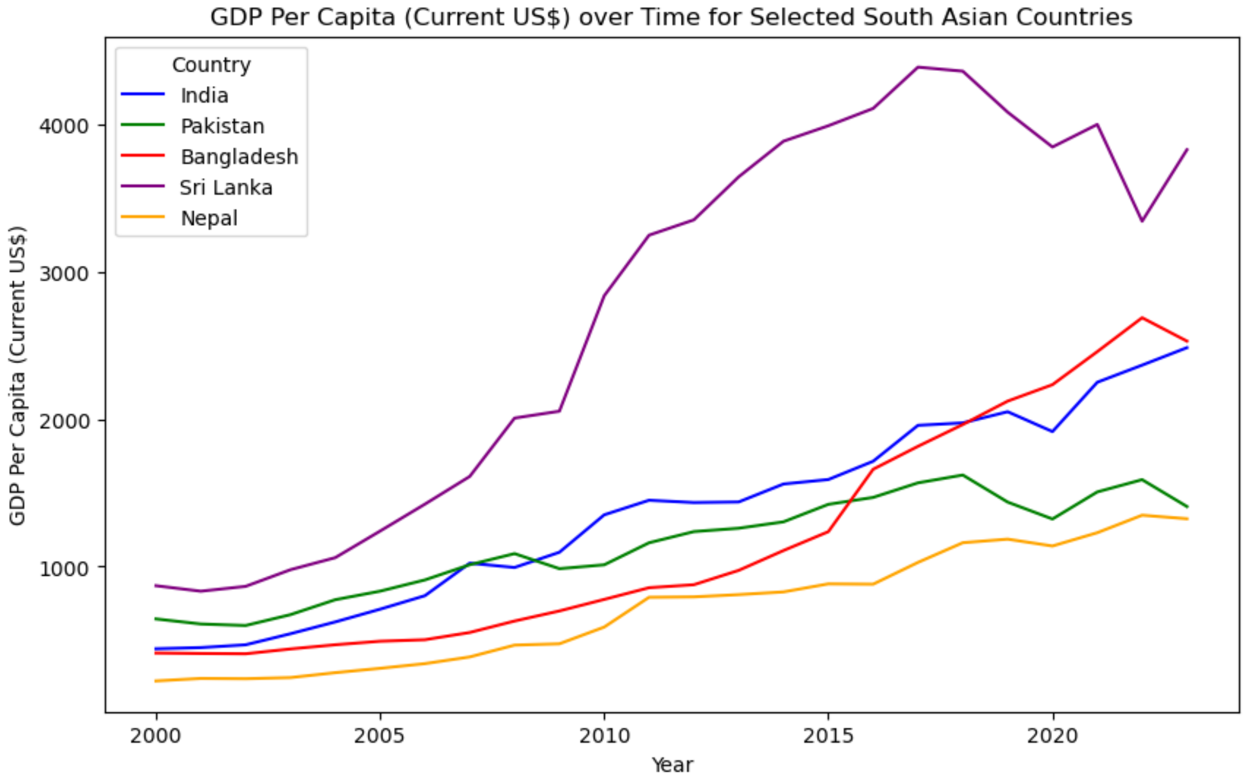
In [64]:

```
# Plotting GDP Growth Rate
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = economic_growth_filtered[economic_growth_filtered["Cou
    plt.plot(country_data["Year"], country_data["GDP growth (annual %)"]
plt.title("GDP Growth Rate (Annual %) over Time for Selected South Asian
plt.xlabel("Year")
plt.ylabel("GDP Growth Rate (Annual %)")
plt.legend(title="Country")
plt.show()
```



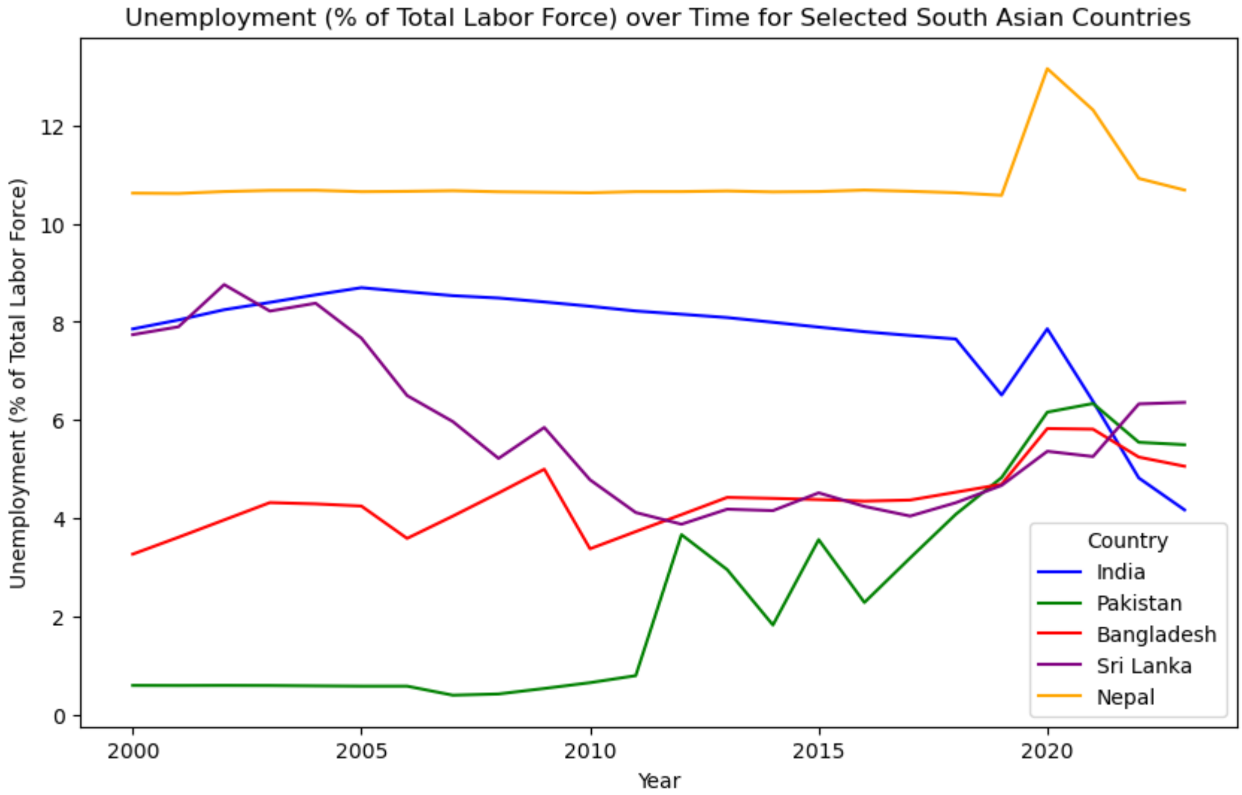
In [65]:

```
# Plotting GDP Per Capita
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = economic_growth_filtered[economic_growth_filtered["Cou
    plt.plot(country_data["Year"], country_data["GDP per capita (current
plt.title("GDP Per Capita (Current US$) over Time for Selected South Asia
plt.xlabel("Year")
plt.ylabel("GDP Per Capita (Current US$)")
plt.legend(title="Country")
plt.show()
```



In [66]:

```
# Plotting Unemployment Rate
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = economic_growth_filtered[economic_growth_filtered["Cou
    plt.plot(country_data["Year"], country_data["Unemployment, total (% o
             label=country, color=country_colors[country])
plt.title("Unemployment (% of Total Labor Force) over Time for Selected S
plt.xlabel("Year")
plt.ylabel("Unemployment (% of Total Labor Force)")
plt.legend(title="Country")
plt.show()
```

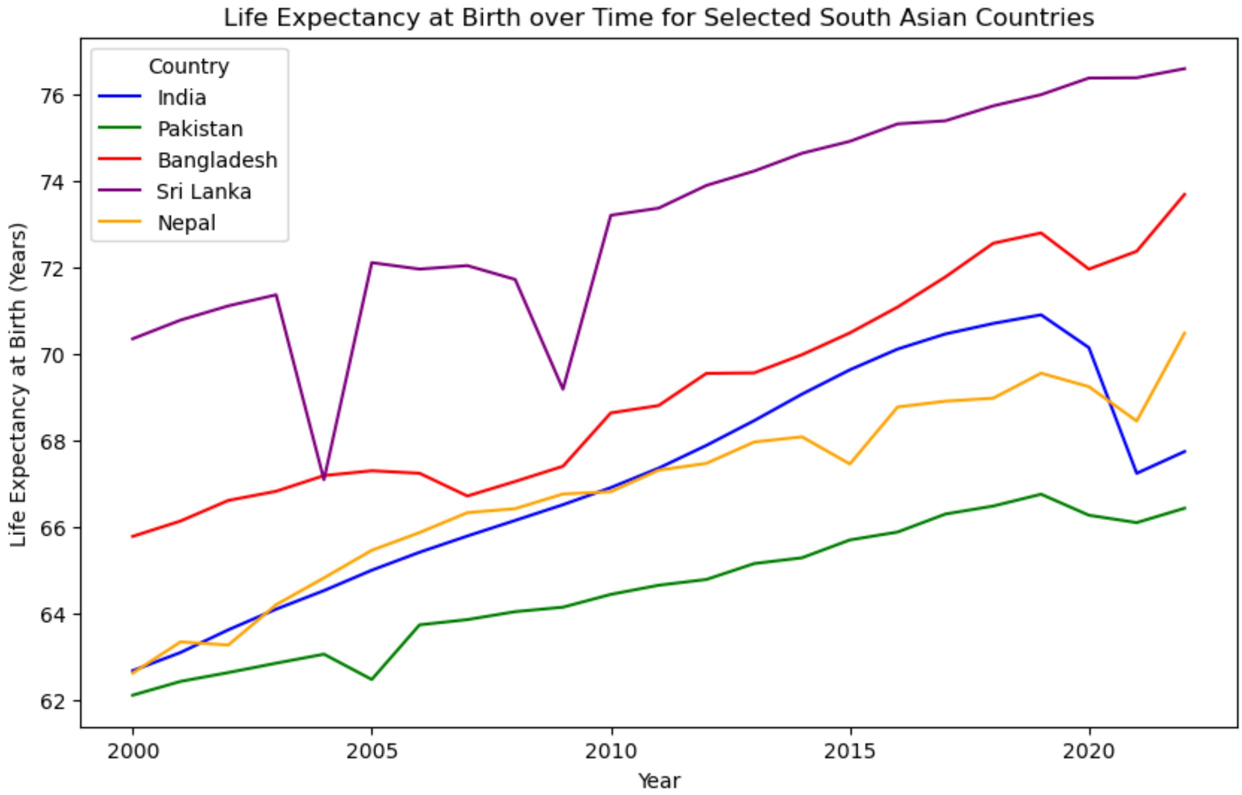


In [67]:

```
# Select relevant columns for social indicators
social_cols = [
    'Country', 'Year', 'Poverty headcount ratio at $2.15 a day (2017 PPP
    "Life expectancy at birth, total (years)", "Mortality rate, infant (
    'Urban population (% of total population)',
    'Individuals using the Internet (% of population)'
]
social_growth_data = data[social_cols]

# Filter the dataset for selected countries
social_growth_filtered = social_growth_data[social_growth_data["Country"
```

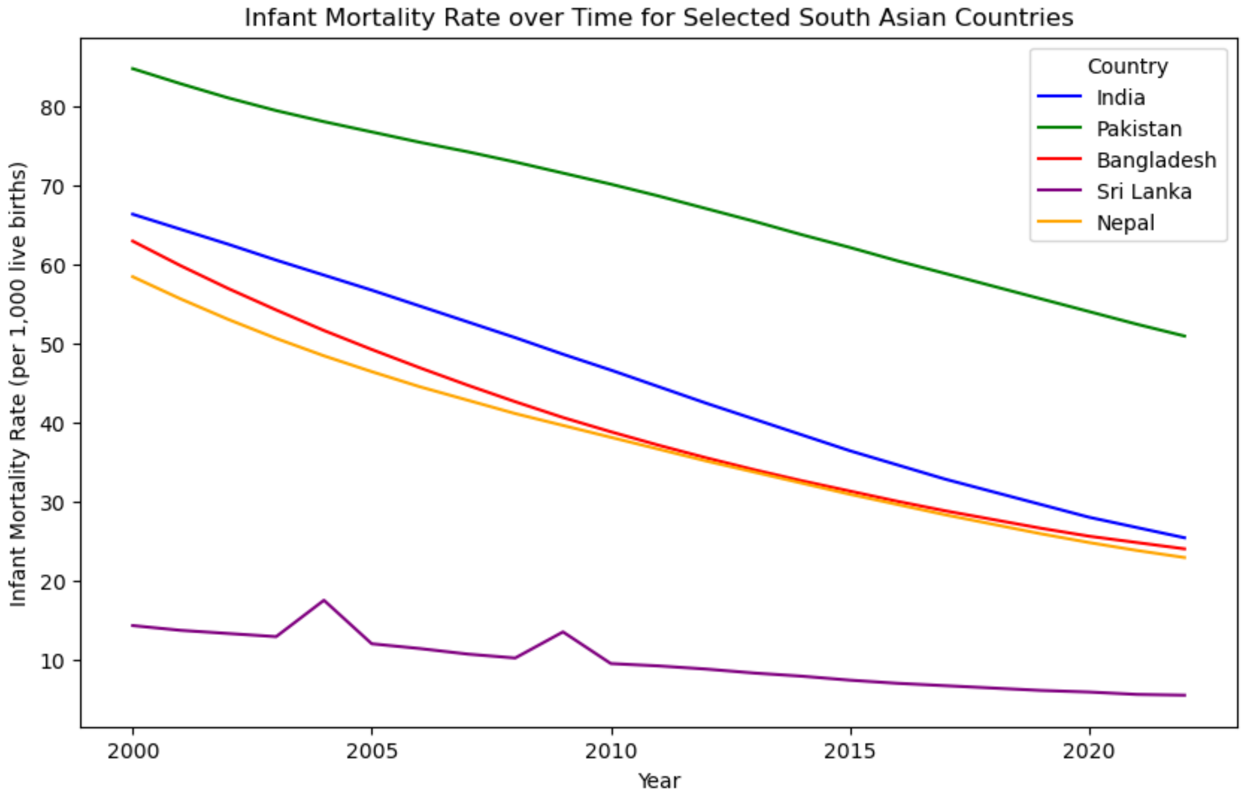
In [68]:

```
# Plotting Life Expectancy at Birth
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = social_growth_filtered[social_growth_filtered["Country
    plt.plot(country_data["Year"], country_data["Life expectancy at birth
             label=country, color=country_colors[country])
plt.title("Life Expectancy at Birth over Time for Selected South Asian Co
plt.xlabel("Year")
plt.ylabel("Life Expectancy at Birth (Years)")
plt.legend(title="Country")
plt.show()
```



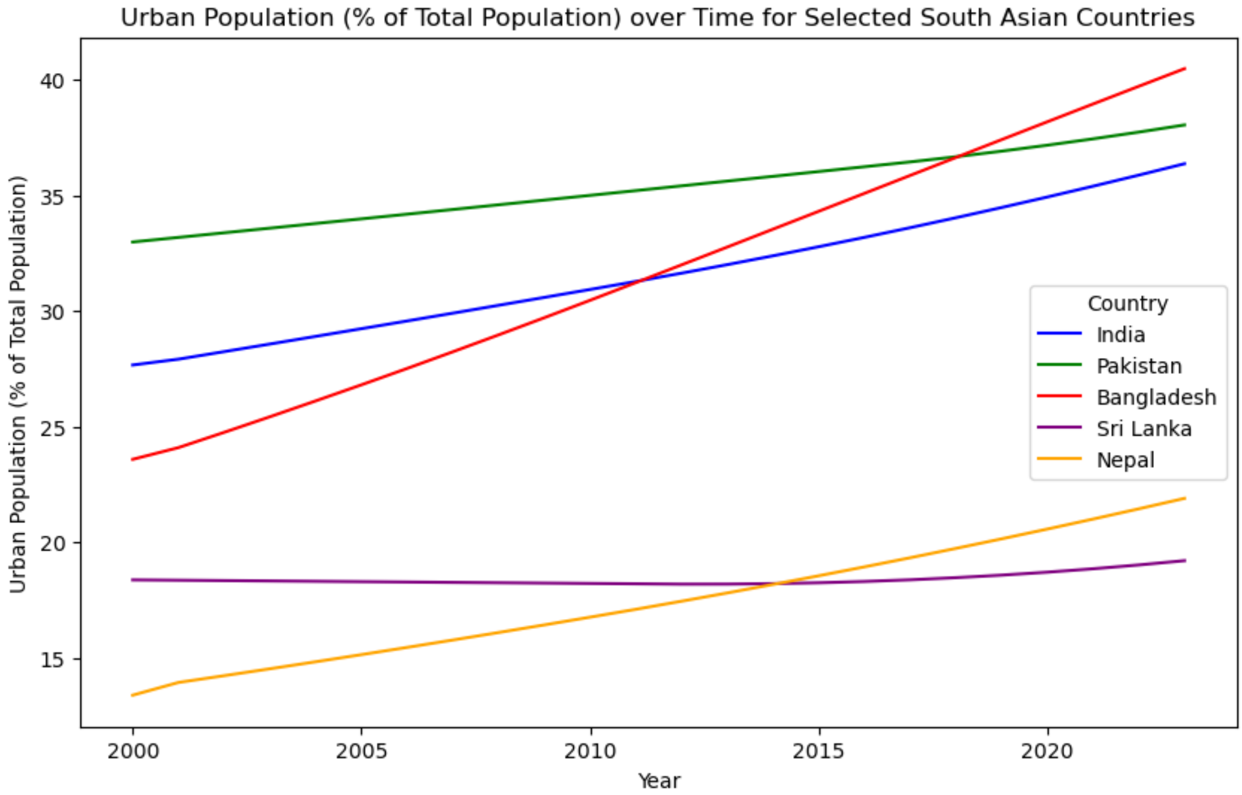
In [69]:

```
# Plotting Infant Mortality Rate
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = social_growth_filtered[social_growth_filtered["Country
    plt.plot(country_data["Year"], country_data["Mortality rate, infant
             label=country, color=country_colors[country])
plt.title("Infant Mortality Rate over Time for Selected South Asian Coun
plt.xlabel("Year")
plt.ylabel("Infant Mortality Rate (per 1,000 live births)")
plt.legend(title="Country")
plt.show()
```



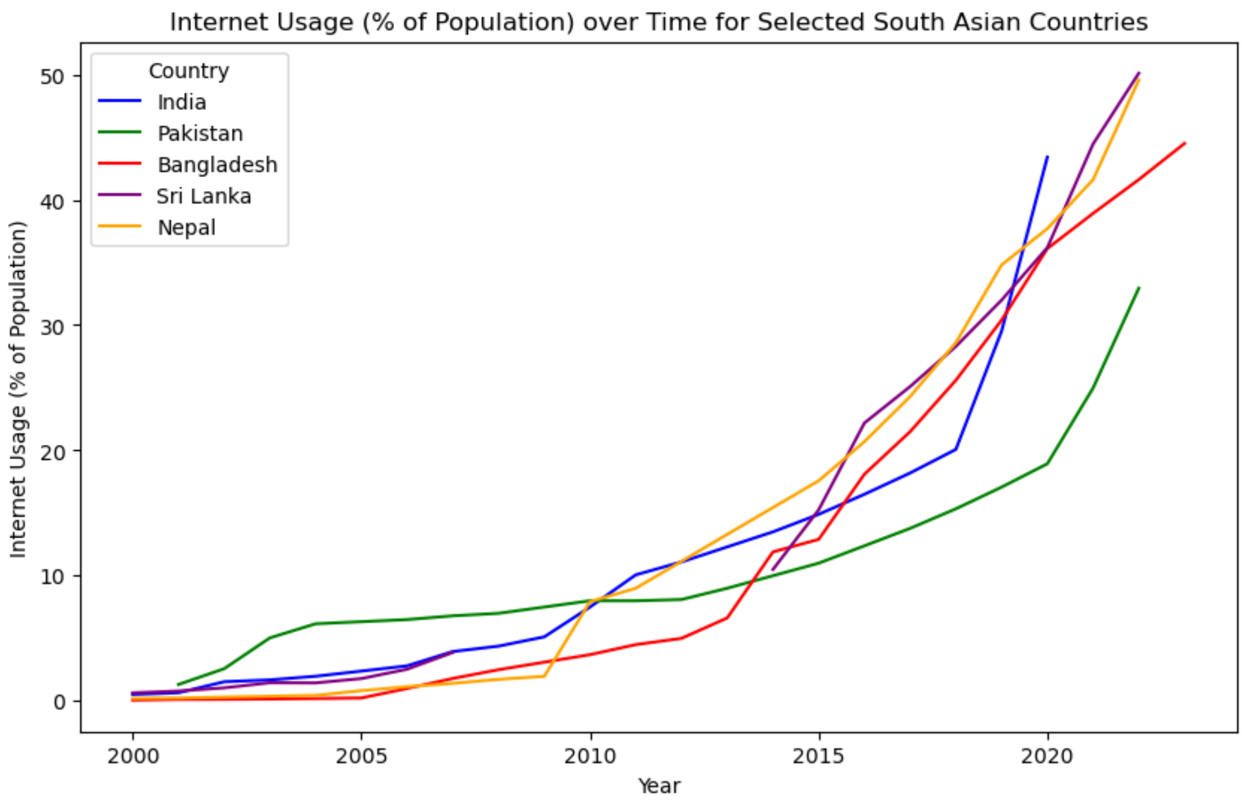
In [70]:

```
# Plotting Urban Population
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = social_growth_filtered[social_growth_filtered["Country
    plt.plot(country_data["Year"], country_data["Urban population (% of t
             label=country, color=country_colors[country])
plt.title("Urban Population (% of Total Population) over Time for Selecte
plt.xlabel("Year")
plt.ylabel("Urban Population (% of Total Population)")
plt.legend(title="Country")
plt.show()
```



In [71]:

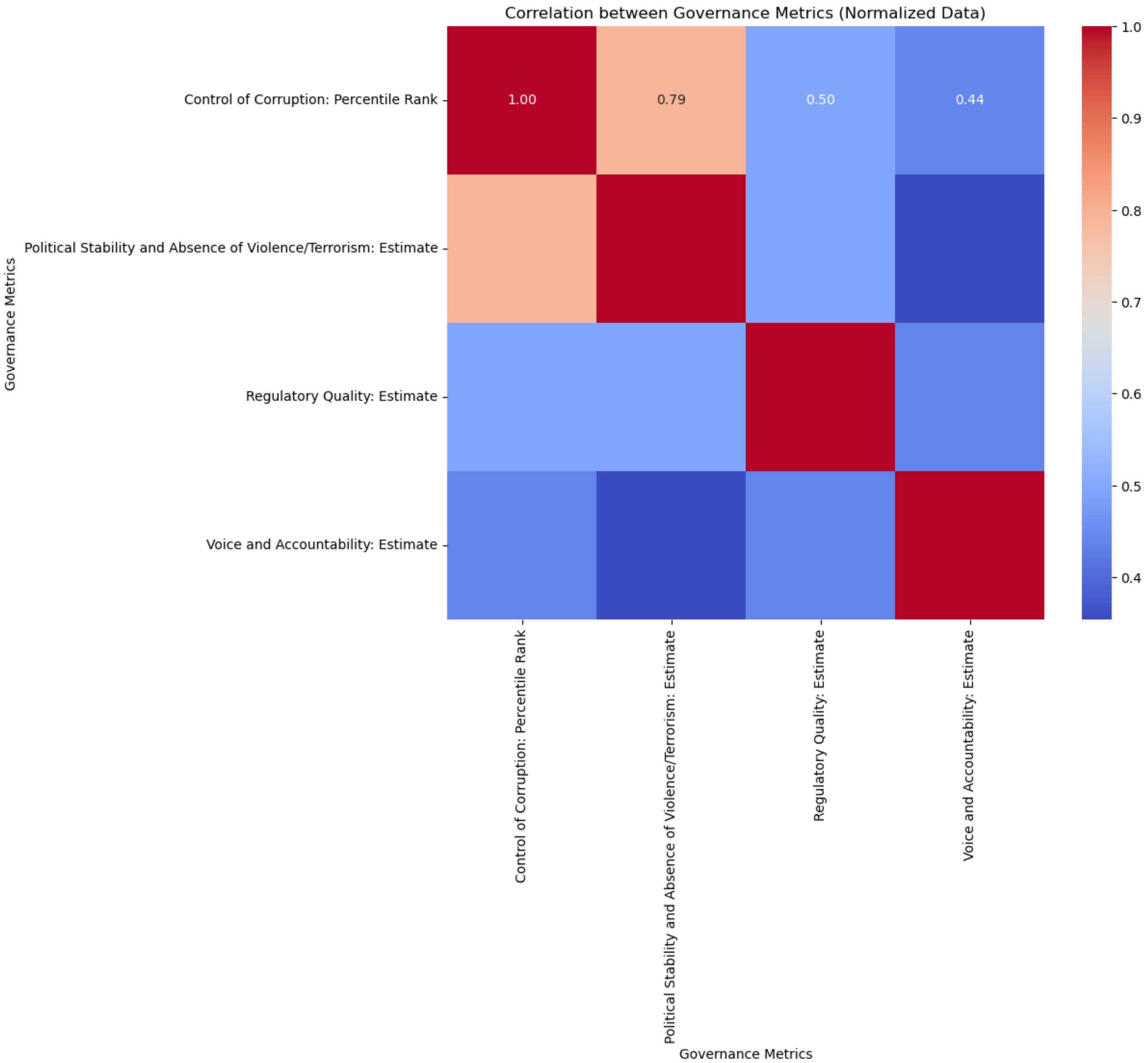
```
# Plotting Internet Usage
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = social_growth_filtered[social_growth_filtered["Country
    plt.plot(country_data["Year"], country_data["Individuals using the In
             label=country, color=country_colors[country])
plt.title("Internet Usage (% of Population) over Time for Selected South
plt.xlabel("Year")
plt.ylabel("Internet Usage (% of Population)")
plt.legend(title="Country")
plt.show()
```



In [72]:

```
#  Governance and Stability Metrics Analysis

from sklearn.preprocessing import StandardScaler

# Select only the governance metrics columns
metrics = data[
    ['Control of Corruption: Percentile Rank',
     'Political Stability and Absence of Violence/Terrorism: Estimate',
     'Regulatory Quality: Estimate',
     'Voice and Accountability: Estimate']
]

# Standardize the metrics
scaler = StandardScaler()
metrics_normalized = scaler.fit_transform(metrics)

# Create a DataFrame from the normalized data
metrics_normalized_df = pd.DataFrame(
    metrics_normalized,
    columns=['Control of Corruption: Percentile Rank',
             'Political Stability and Absence of Violence/Terrorism: Est
             'Regulatory Quality: Estimate',
             'Voice and Accountability: Estimate']
)

# Calculate correlation on the normalized data
normalized_corr = metrics_normalized_df.corr()

# Plot the heatmap for normalized data
plt.figure(figsize=(10, 8))
sns.heatmap(normalized_corr, annot=True, cmap='coolwarm', fmt=".2f")
```

```
plt.title("Correlation between Governance Metrics (Normalized Data)")
plt.xlabel("Governance Metrics")
plt.ylabel("Governance Metrics")
plt.show()
```

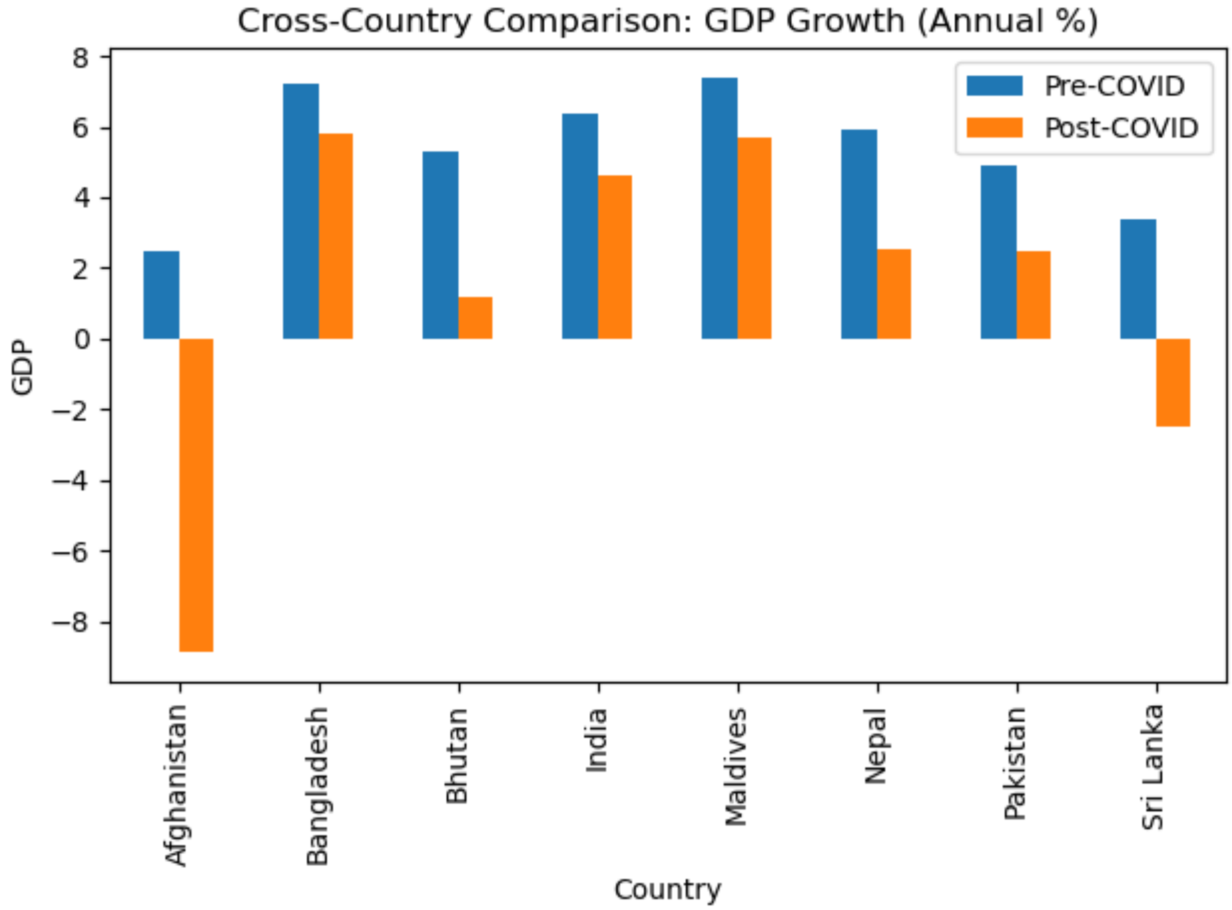


In [73]:

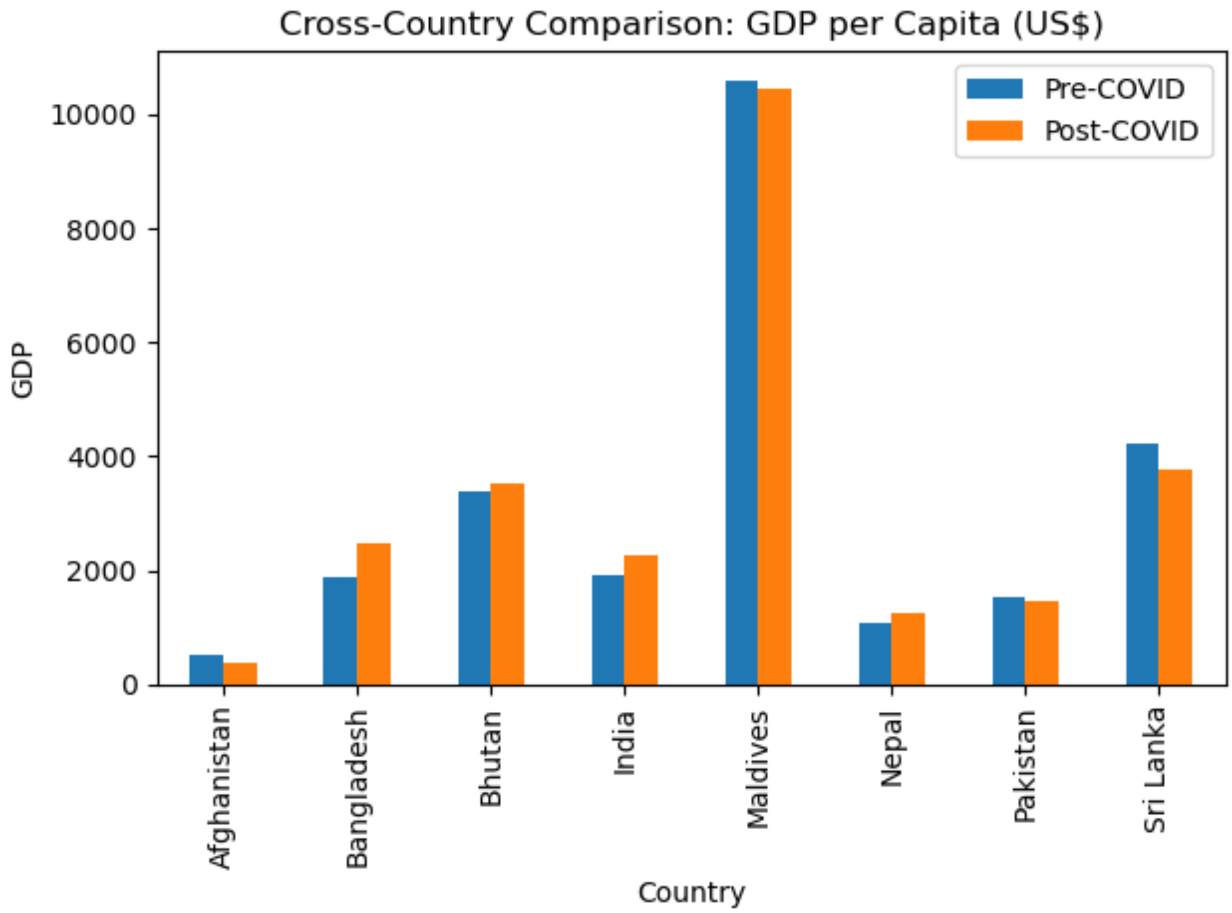
```
# Set options for efficient data manipulation
pd.options.mode.copy_on_write = True

# Data Filtering and Preparation
data_filtered = data[['Country', 'Year', 'GDP growth (annual %)', 'GDP pe
                      'Unemployment, total (% of total labor force) (mode
                      'Inflation, consumer prices (annual %)', 'School en

# Sort and fill NaN values by forward and backward filling within each co
data_filtered.sort_values(by=['Country', 'Year'], inplace=True)
data_filtered.ffill(inplace=True)
data_filtered.bfill(inplace=True)

# Split data into pre-COVID (2016-2019) and post-COVID (2020-2023) for co
pre_covid_data = data_filtered[(data_filtered['Year'] >= 2016) & (data_fi
```

```
post_covid_data = data_filtered[data_filtered['Year'] >= 2020]

# Aggregate means by country for each period
pre_covid_avg = pre_covid_data.groupby('Country').mean().reset_index()
post_covid_avg = post_covid_data.groupby('Country').mean().reset_index()

# Merging the pre and post-COVID averages for comparative visualization
comparison_df = pd.merge(pre_covid_avg, post_covid_avg, on='Country', su

# Indicators to plot individually
indicators = [
    ('GDP growth (annual %)_pre_covid', 'GDP growth (annual %)_post_covi
    ('GDP per capita (current US$)_pre_covid', 'GDP per capita (current U
    ('Unemployment, total (% of total labor force) (modeled ILO estimate
     'Unemployment, total (% of total labor force) (modeled ILO estimate
    ('Inflation, consumer prices (annual %)_pre_covid', 'Inflation, consu
    ('School enrollment, primary (% gross)_pre_covid', 'School enrollment
]

# Loop through each indicator to create individual plots
for pre_col, post_col, title in indicators:
    plt.figure(figsize=(12, 6))
    comparison_df.plot(kind='bar', x='Country', y=[pre_col, post_col])
    plt.title(f"Cross-Country Comparison: {title}")
    plt.xlabel("Country")
    plt.ylabel(title.split()[0])
    plt.legend(["Pre-COVID", "Post-COVID"])
    plt.tight_layout()
    plt.show()
```

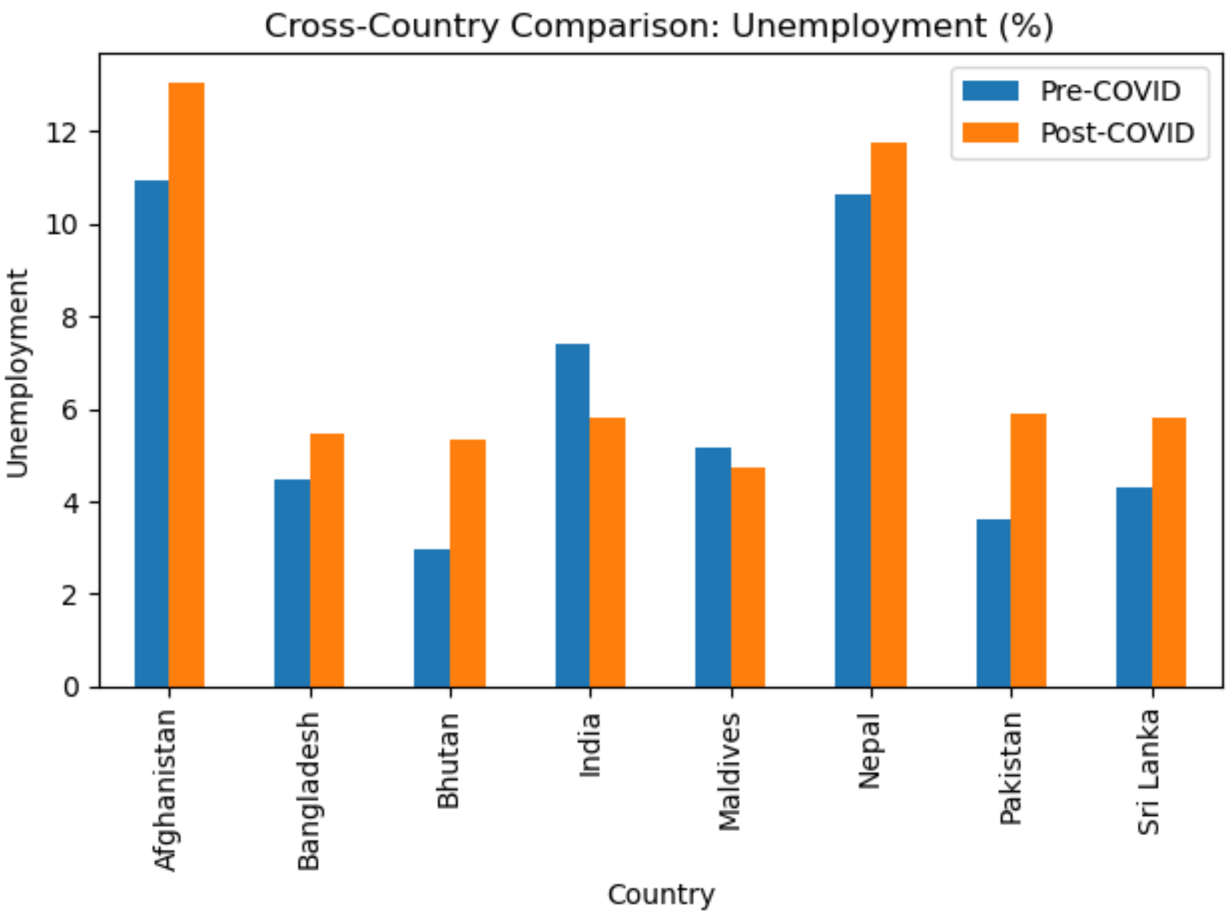
```
<Figure size 1200x600 with 0 Axes>
```



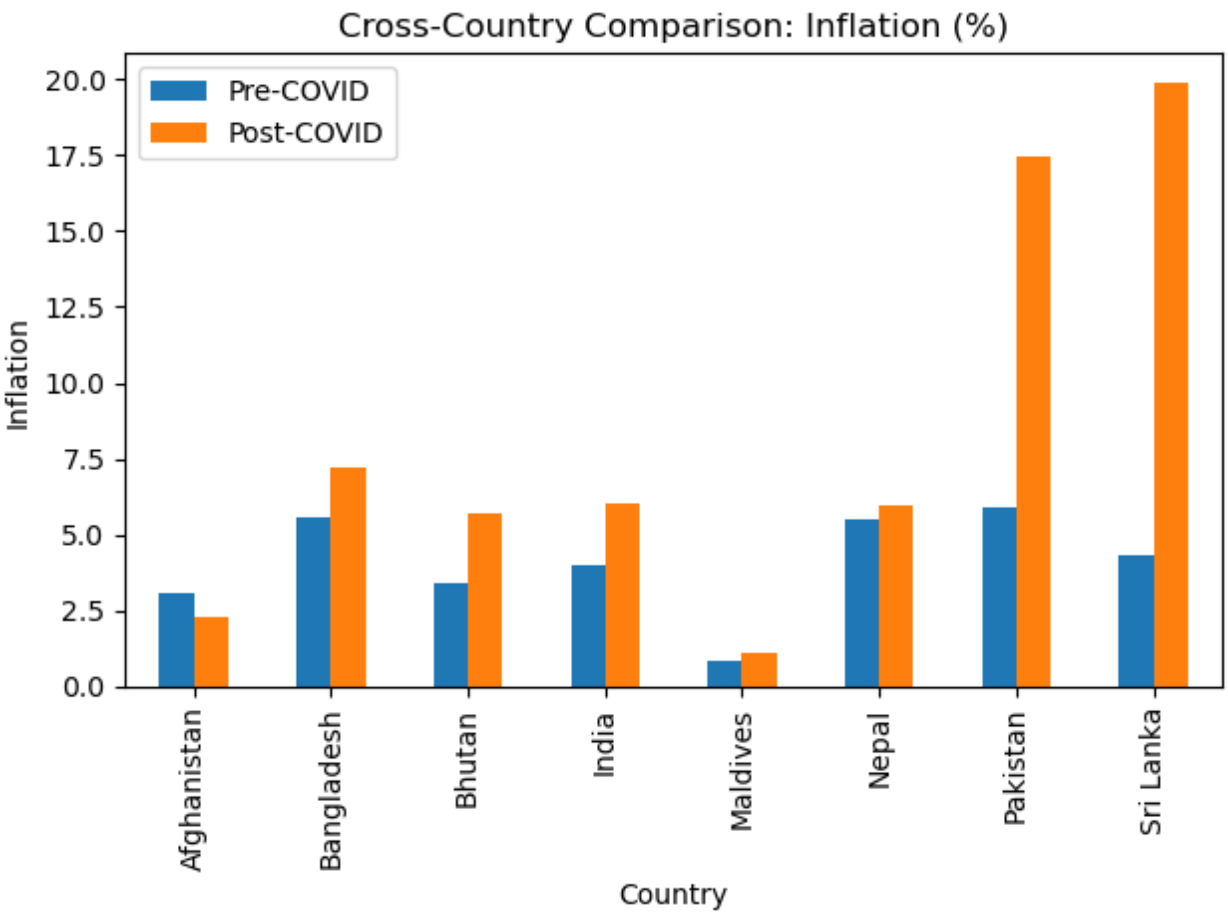
<Figure size 1200x600 with 0 Axes>



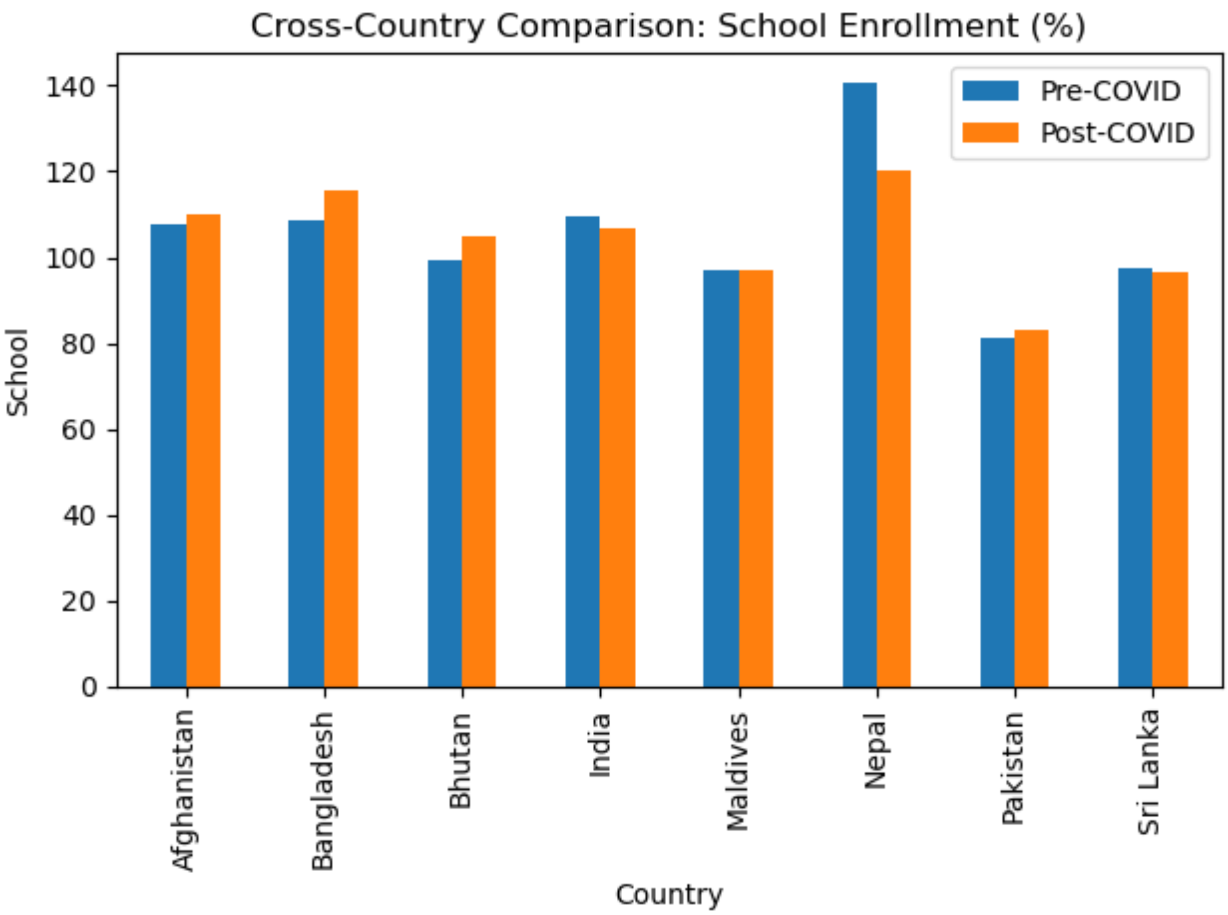
<Figure size 1200x600 with 0 Axes>



<Figure size 1200x600 with 0 Axes>

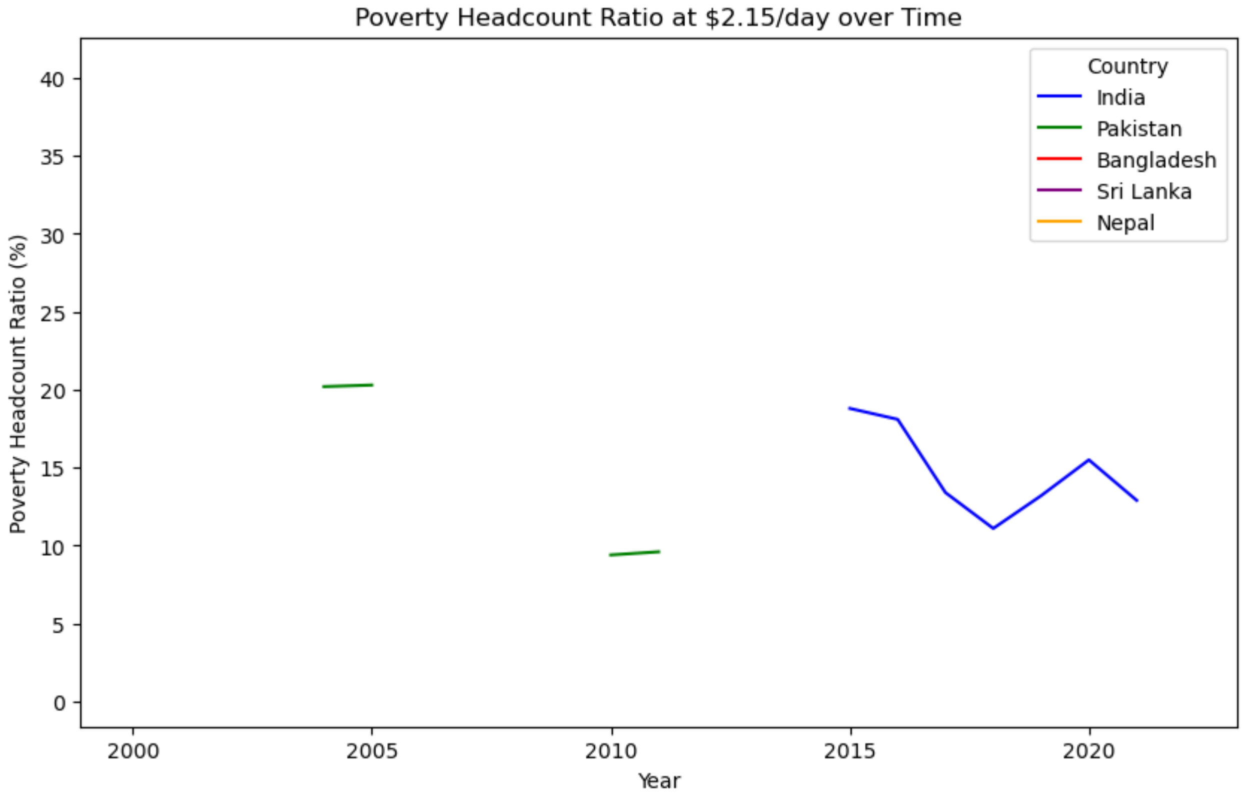


<Figure size 1200x600 with 0 Axes>



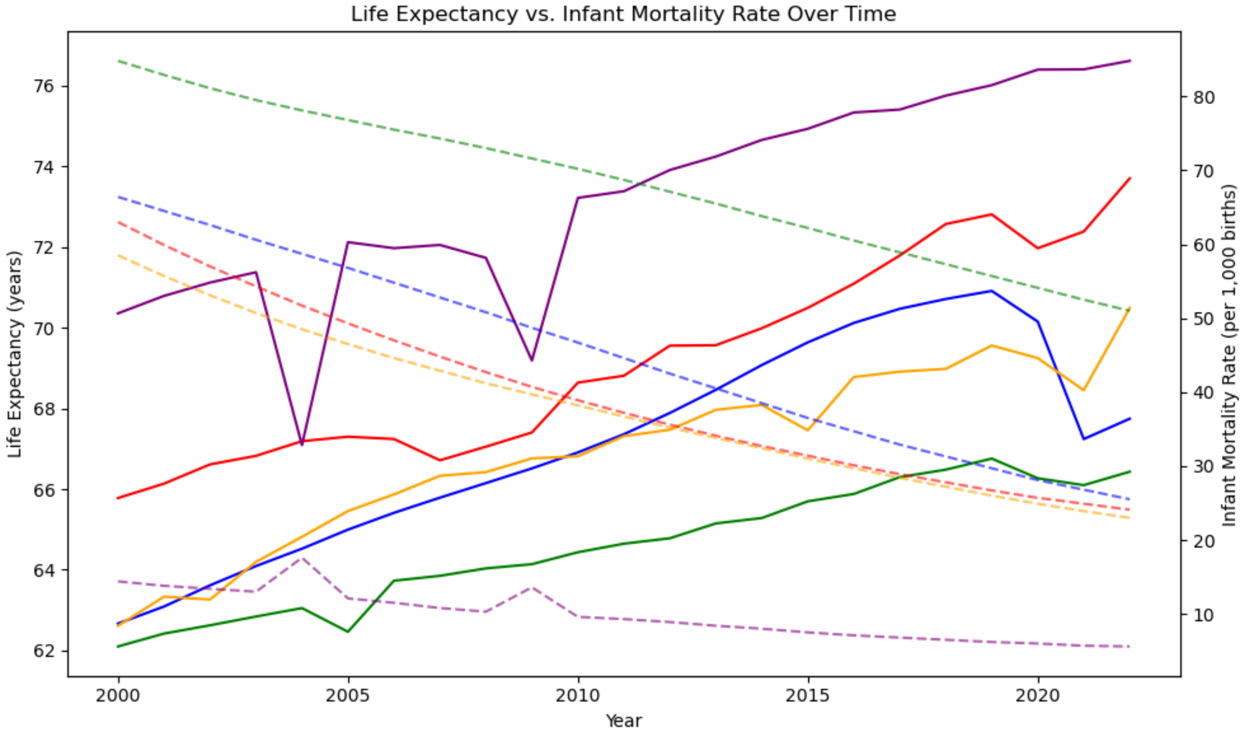
In [74]:

```
# Poverty Headcount Ratio Over Time
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = social_growth_filtered[social_growth_filtered["Country
    plt.plot(country_data["Year"], country_data["Poverty headcount ratio
             label=country, color=country_colors[country])
plt.title("Poverty Headcount Ratio at $2.15/day over Time")
plt.xlabel("Year")
plt.ylabel("Poverty Headcount Ratio (%)")
plt.legend(title="Country")
plt.show()
```

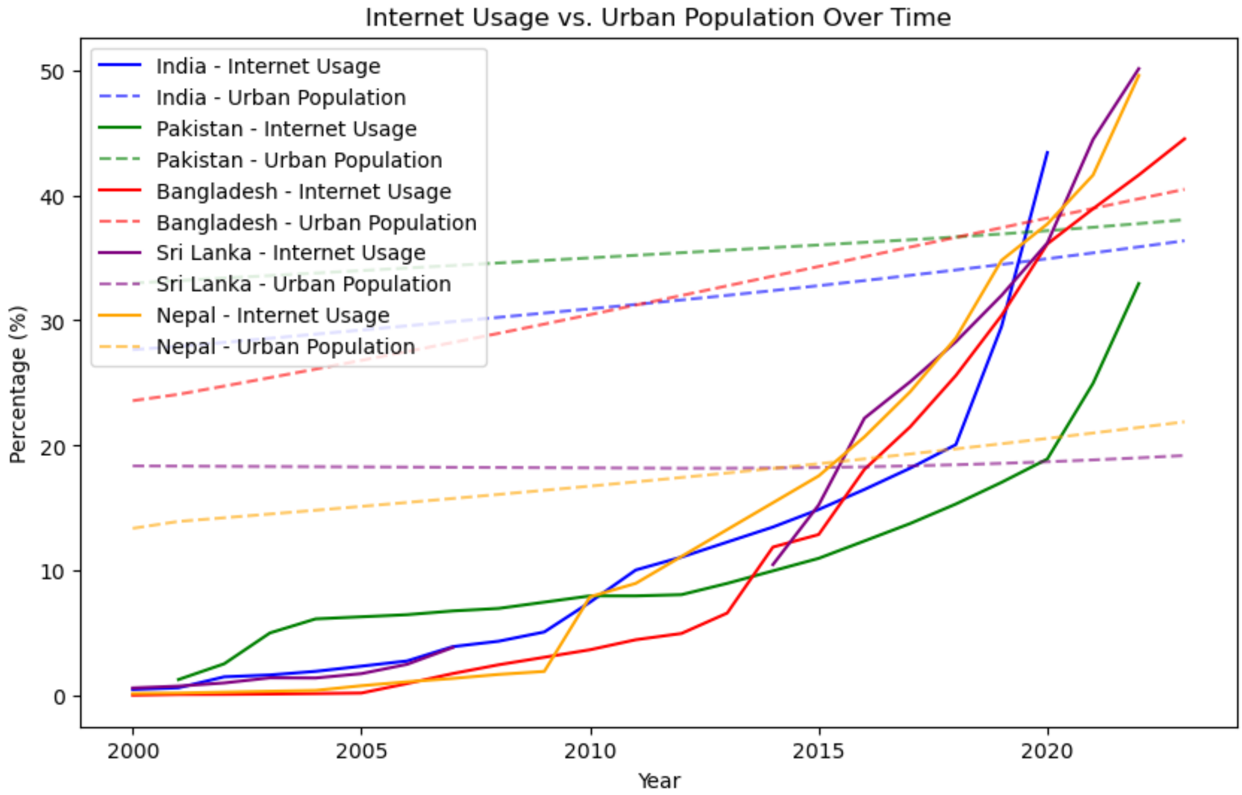


In [75]:

```
# Life Expectancy and Infant Mortality (Dual-axis Plot)
fig, ax1 = plt.subplots(figsize=(10, 6))
ax2 = ax1.twinx()
for country in selected_countries:
    country_data = social_growth_filtered[social_growth_filtered["Countr
    ax1.plot(country_data["Year"], country_data["Life expectancy at birt
    ax2.plot(country_data["Year"], country_data["Mortality rate, infant
ax1.set_xlabel("Year")
ax1.set_ylabel("Life Expectancy (years)")
ax2.set_ylabel("Infant Mortality Rate (per 1,000 births)")
plt.title("Life Expectancy vs. Infant Mortality Rate Over Time")
fig.tight_layout()
plt.show()
```

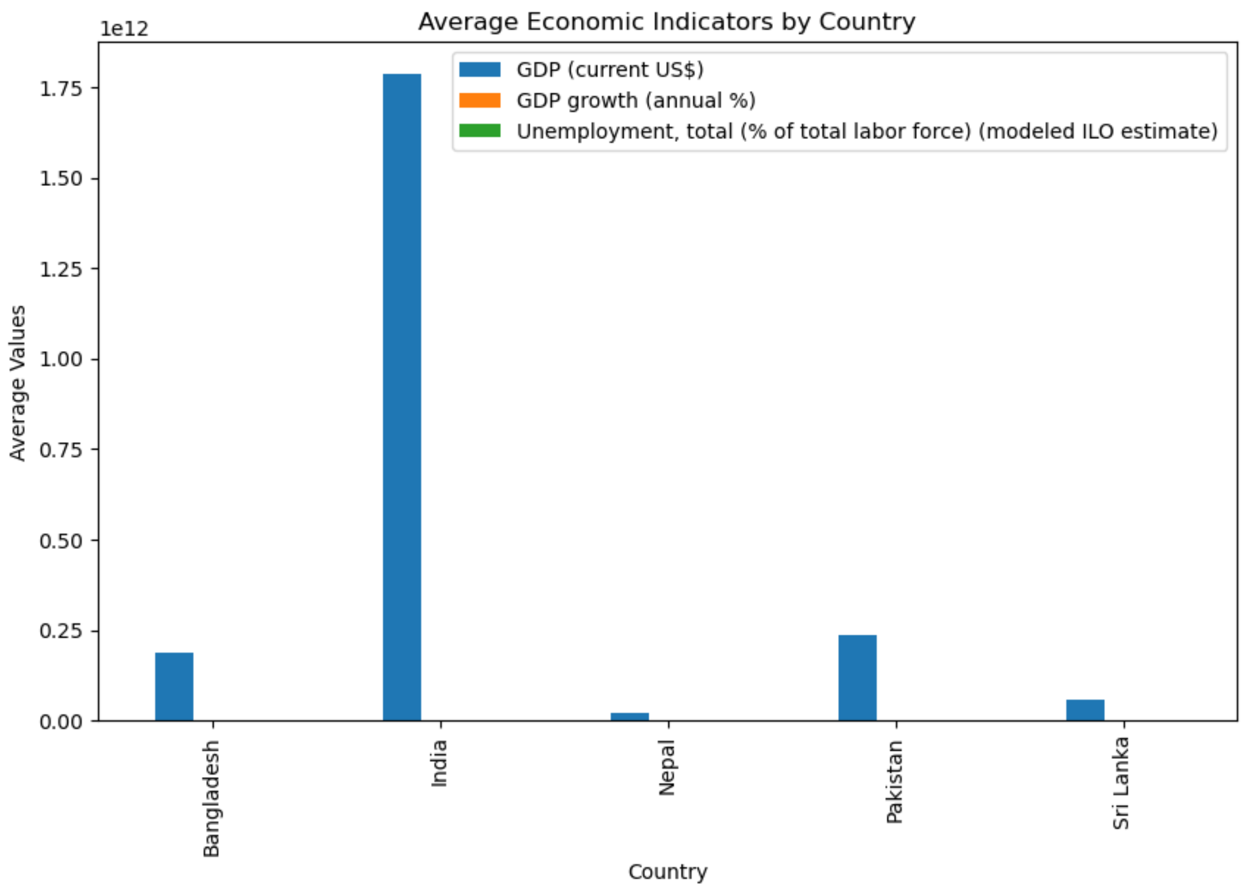


In [76]:
```
# Internet Usage vs. Urban Population
plt.figure(figsize=(10, 6))
for country in selected_countries:
    country_data = social_growth_filtered[social_growth_filtered["Country
    plt.plot(country_data["Year"], country_data["Individuals using the In
    plt.plot(country_data["Year"], country_data["Urban population (% of t
plt.title("Internet Usage vs. Urban Population Over Time")
plt.xlabel("Year")
plt.ylabel("Percentage (%)")
plt.legend()
plt.show()
```



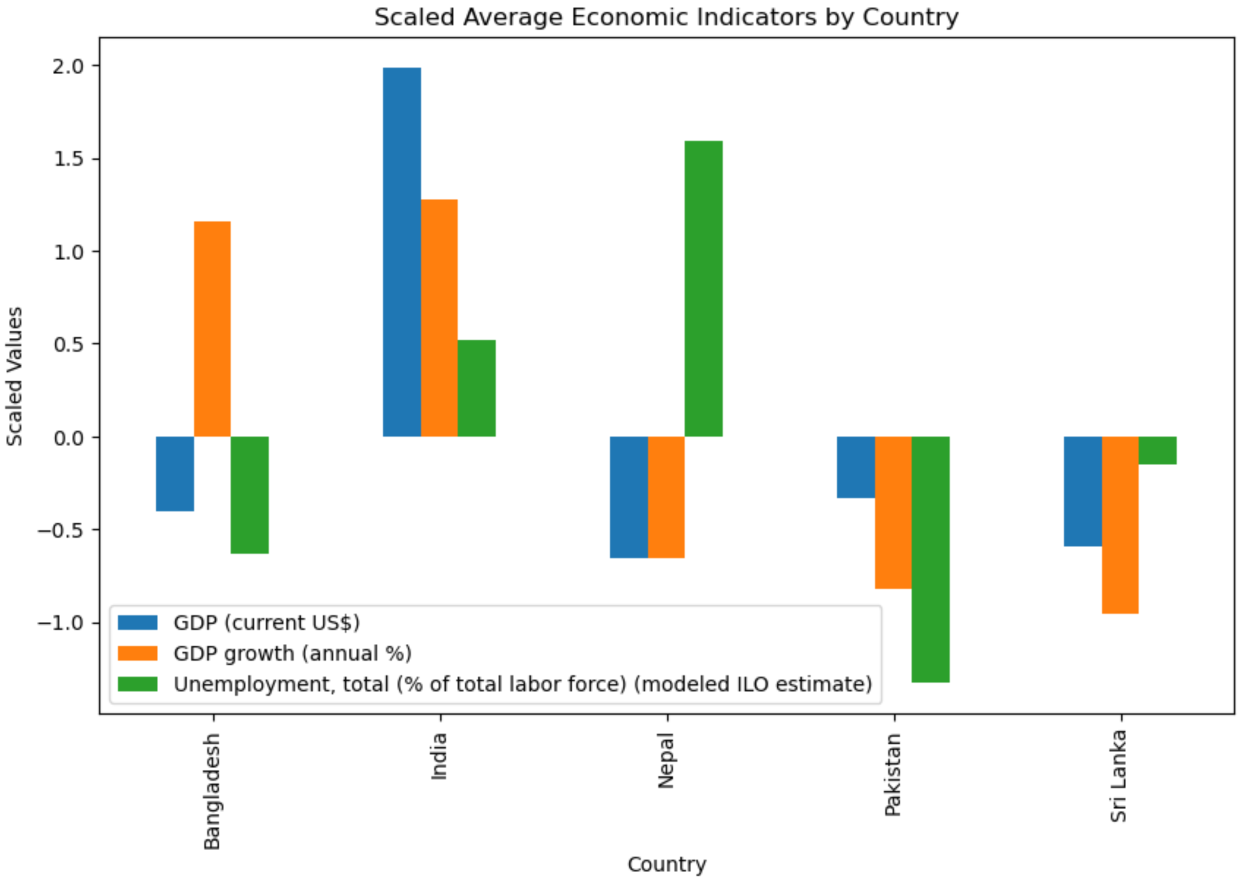
In [77]:

```
# Average Economic Indicators by Country
avg_economic = economic_growth_filtered.groupby("Country").mean()
avg_economic[["GDP (current US$)", "GDP growth (annual %)", "Unemploymen
plt.title("Average Economic Indicators by Country")
plt.ylabel("Average Values")
plt.show()
```



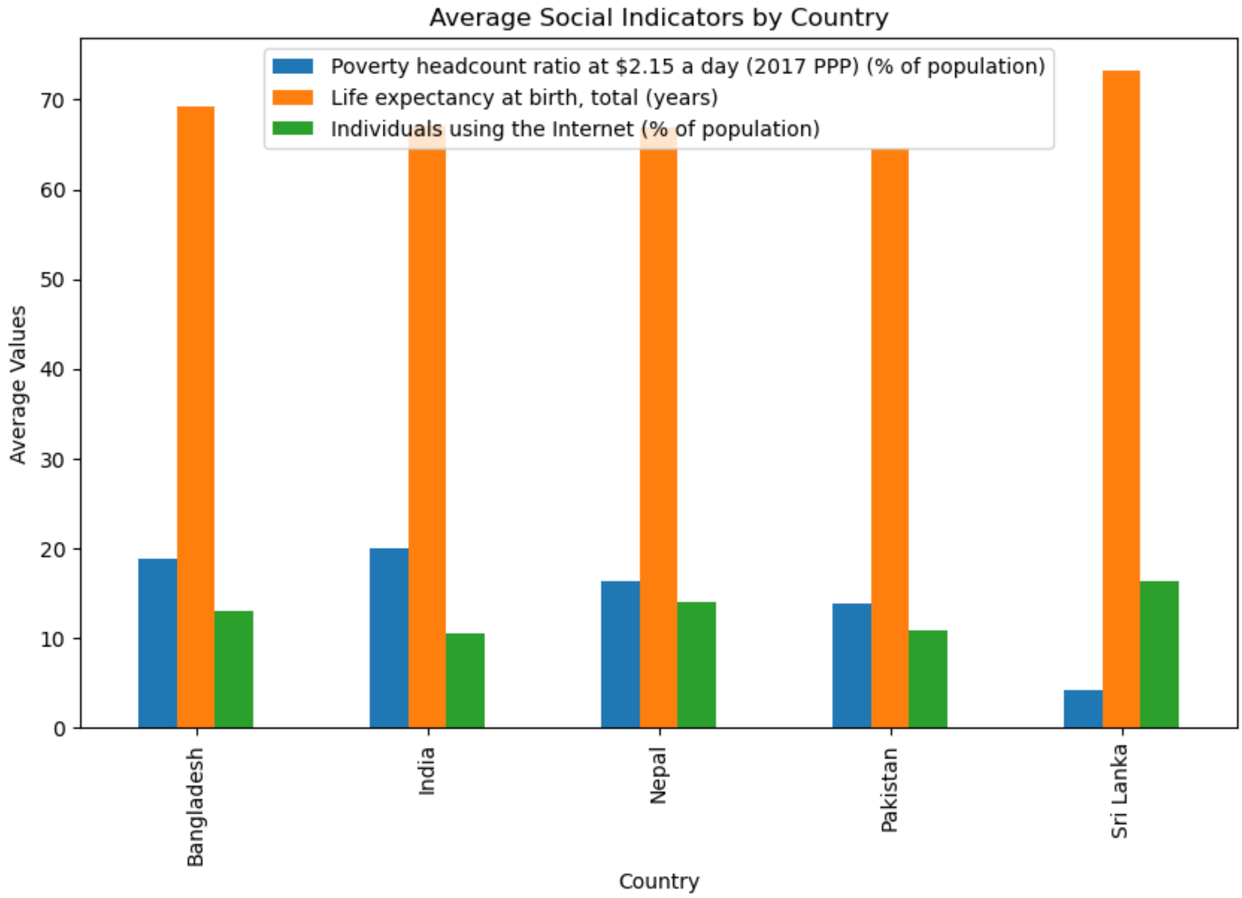
In [78]:

```
from sklearn.preprocessing import StandardScaler

# Calculate the average values by country
avg_economic = economic_growth_filtered.groupby("Country").mean()

# Select the columns to scale
economic_features = avg_economic[["GDP (current US$)","GDP growth (annua

# Scale the selected columns
scaler = StandardScaler()
scaled_features = scaler.fit_transform(economic_features)

# Convert scaled features back to a DataFrame with the same index and co
scaled_df = pd.DataFrame(scaled_features, index=avg_economic.index, colu

# Plot the scaled data
scaled_df.plot(kind="bar", figsize=(10, 6))
plt.title("Scaled Average Economic Indicators by Country")
plt.ylabel("Scaled Values")
plt.show()
```



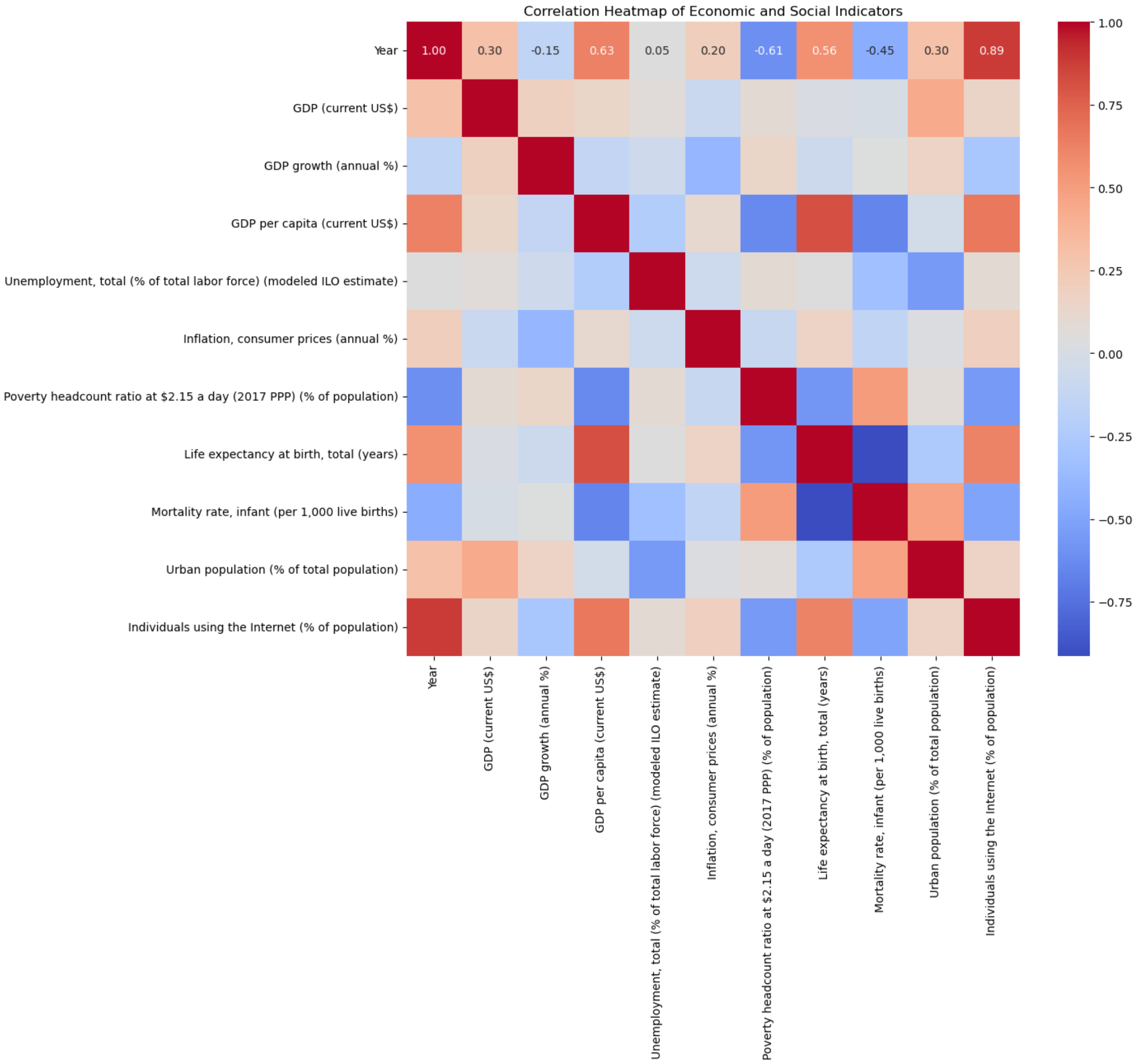
In [79]:

```
# Average Social Indicators by Country
avg_social = social_growth_filtered.groupby("Country").mean()
avg_social[["Poverty headcount ratio at $2.15 a day (2017 PPP) (% of popu
plt.title("Average Social Indicators by Country")
plt.ylabel("Average Values")
plt.show()
```



In [80]:

```
# Merge data and drop non-numeric columns
correlation_data = economic_growth_filtered.merge(social_growth_filtered

# Drop non-numeric columns
correlation_data_numeric = correlation_data.select_dtypes(include=[float

# Plot the heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_data_numeric.corr(), annot=True, cmap="coolwarm"
plt.title("Correlation Heatmap of Economic and Social Indicators")
plt.show()
```



In [81]:

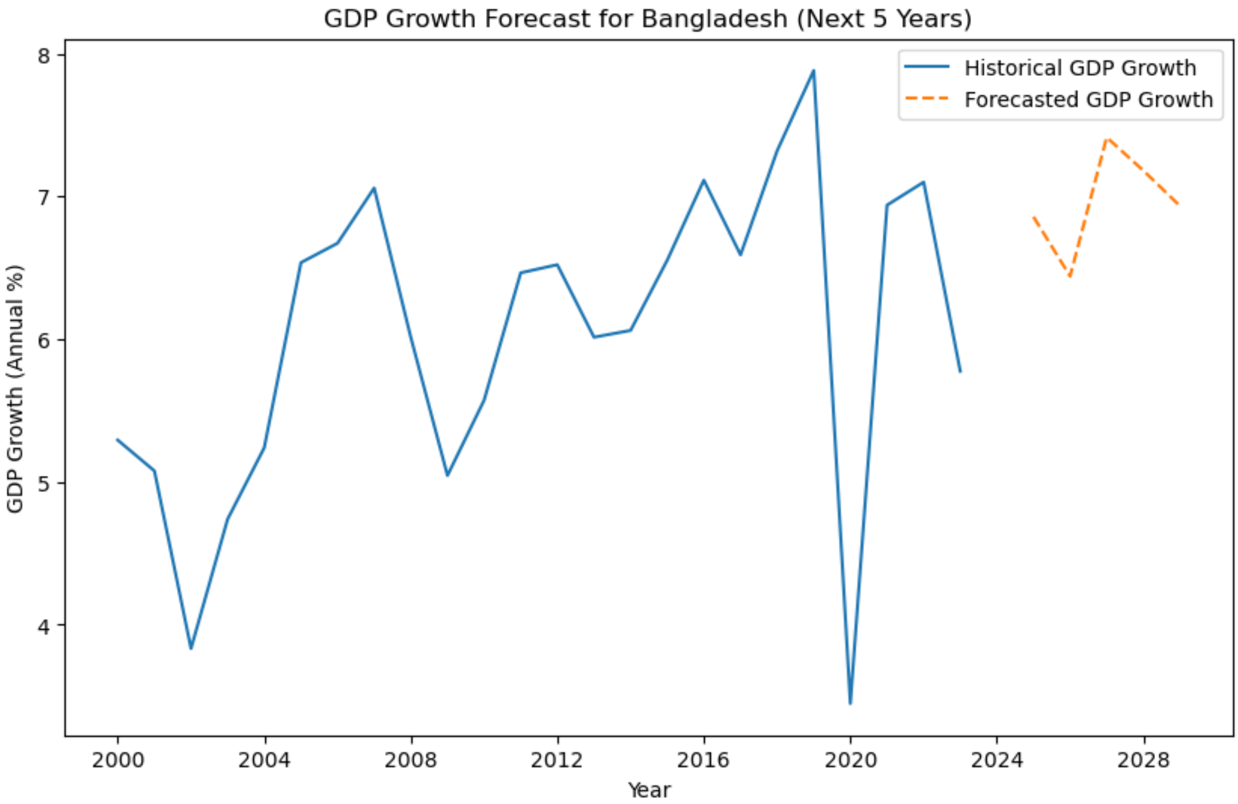
```
data.columns
```

Out[81]:
```
Index(['Country', 'Year', 'GDP (current US$)', 'GDP growth (annual %)',
       'GDP per capita (current US$)',
       'Unemployment, total (% of total labor force) (modeled ILO estima
te)',
       'Inflation, consumer prices (annual %)',
       'Foreign direct investment, net inflows (% of GDP)', 'Trade (% of
GDP)',
       'Gini index', 'Population, total', 'Population growth (annual %)'
,
       'Poverty headcount ratio at $2.15 a day (2017 PPP) (% of populati
on)',
       'Life expectancy at birth, total (years)',
       'Mortality rate, infant (per 1,000 live births)',
       'Literacy rate, adult total (% of people ages 15 and above)',
       'School enrollment, primary (% gross)',
       'Urban population (% of total population)',
       'Access to electricity (% of population)',
       'People using at least basic drinking water services (% of popula
tion)',
       'People using at least basic sanitation services (% of population
)',
       'Carbon dioxide (CO2) emissions excluding LULUCF per capita (t CO
2e/capita)',
       'PM2.5 air pollution, mean annual exposure (micrograms per cubic
meter)',
       'Renewable energy consumption (% of total final energy consumptio
n)',
       'Forest area (% of land area)',
       'Control of Corruption: Percentile Rank',
       'Political Stability and Absence of Violence/Terrorism: Estimate'
,
       'Regulatory Quality: Estimate', 'Rule of Law: Estimate',
       'Voice and Accountability: Estimate',
       'Individuals using the Internet (% of population)'],
      dtype='object')
```

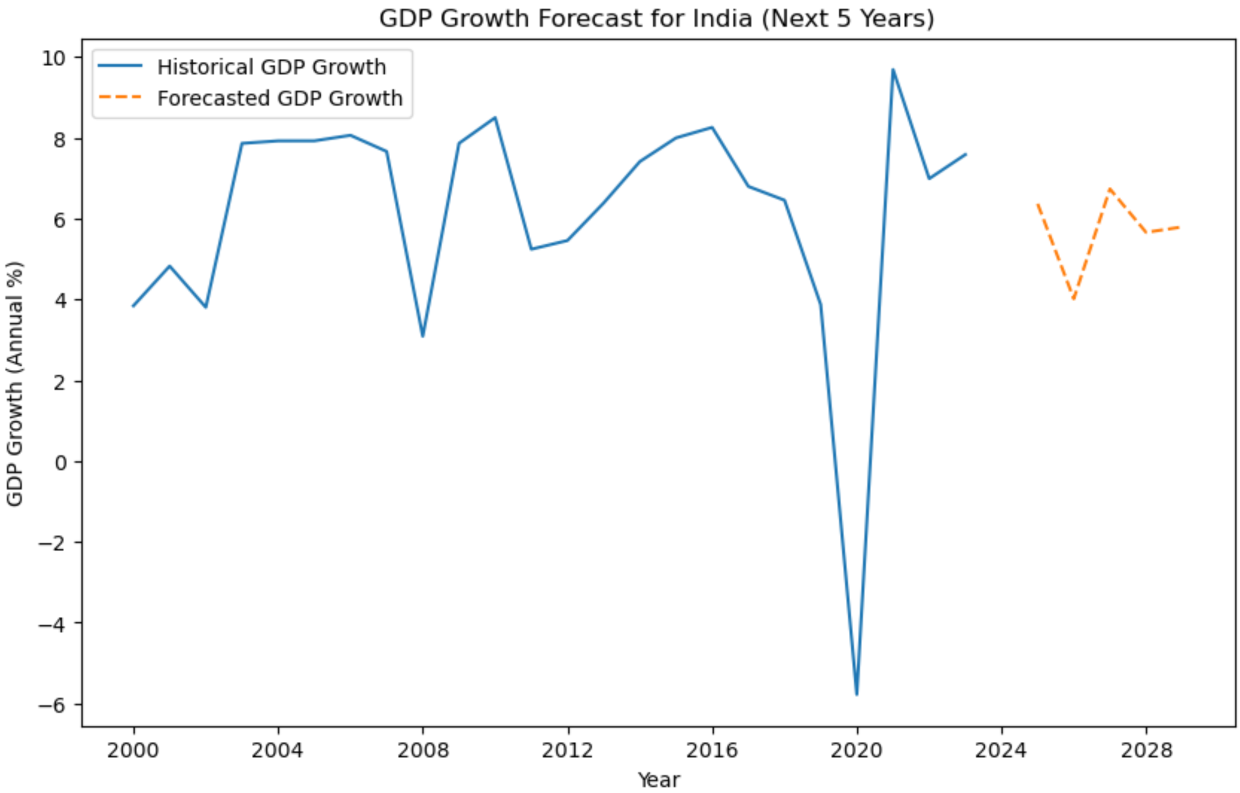
In [82]:
```python
# List of countries for which you want to generate the forecast
countries = economic_growth_filtered["Country"].unique()

# Dictionary to store forecasts for each country
country_forecasts = {}

# Loop through each country
for country in countries:
    # Filter data for the current country and convert "Year" to datetime
    country_data = economic_growth_filtered[economic_growth_filtered["Cou
    country_data["Year"] = pd.to_datetime(country_data["Year"], format='
    country_data.set_index("Year", inplace=True)

    # Ensure there is enough data to fit the model
    if country_data["GDP growth (annual %)"].dropna().shape[0] >= 2:
        # Fit the Exponential Smoothing model
        model = ExponentialSmoothing(country_data["GDP growth (annual %)
        fitted_model = model.fit()

        # Forecast the next 5 years
```

```
            forecast = fitted_model.forecast(steps=5)

            # Set forecast index to continue from the last year in the data
            forecast_index = pd.date_range(start=country_data.index[-1] + pd
            forecast.index = forecast_index

            # Store the forecast in the dictionary
            country_forecasts[country] = forecast

            # Plot the historical and forecasted data
            plt.figure(figsize=(10, 6))
            plt.plot(country_data.index, country_data["GDP growth (annual %)
            plt.plot(forecast.index, forecast, label="Forecasted GDP Growth"
            plt.title(f"GDP Growth Forecast for {country} (Next 5 Years)")
            plt.xlabel("Year")
            plt.ylabel("GDP Growth (Annual %)")
            plt.legend()
            plt.show()
        else:
            print(f"Not enough data to forecast for {country}")
```

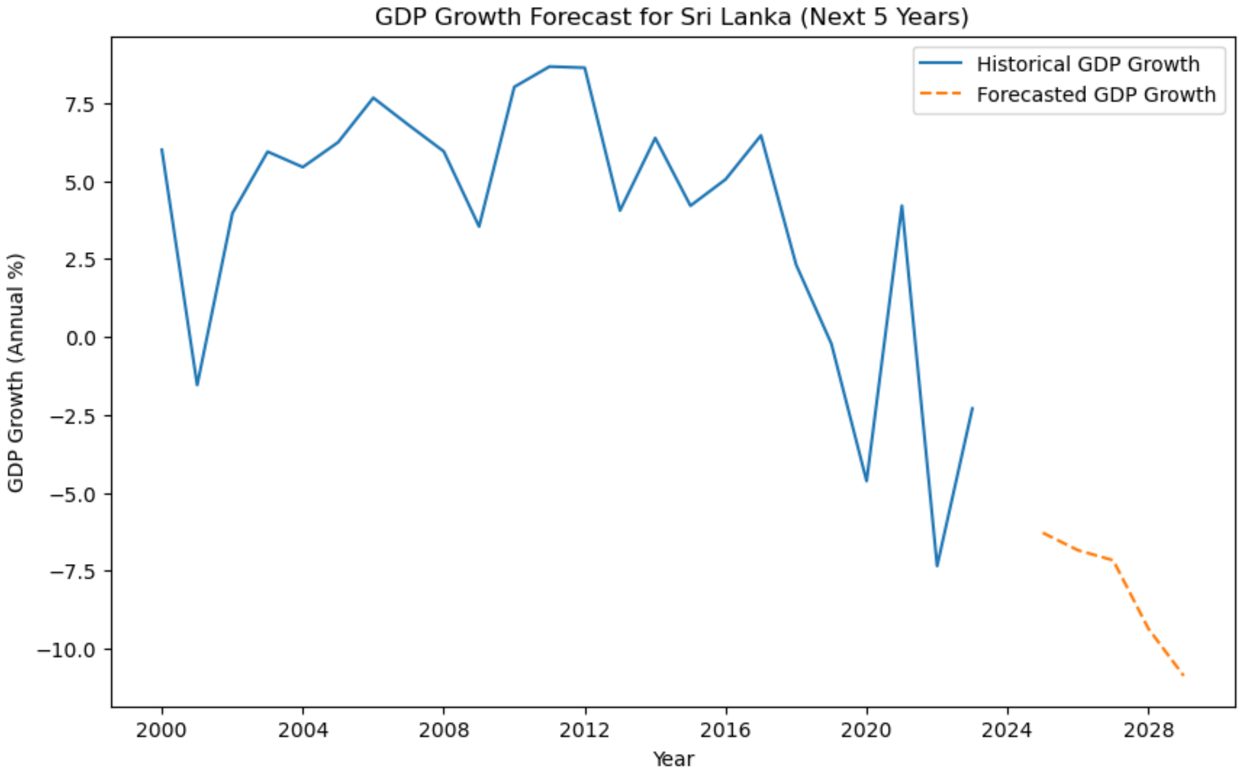
```
C:\Anaconda\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Value
Warning:

No frequency information was provided, so inferred frequency AS-JAN will b
e used.
```



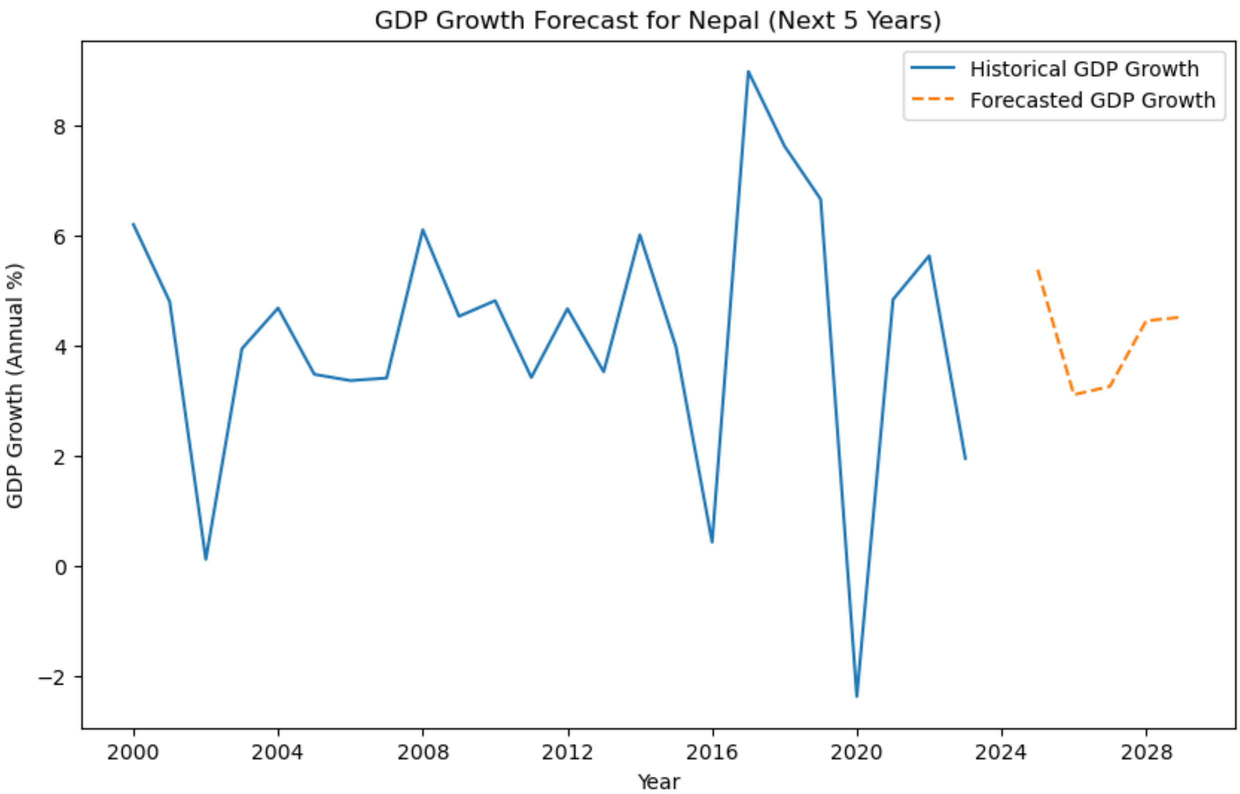
```
C:\Anaconda\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Value
Warning:

No frequency information was provided, so inferred frequency AS-JAN will b
e used.
```



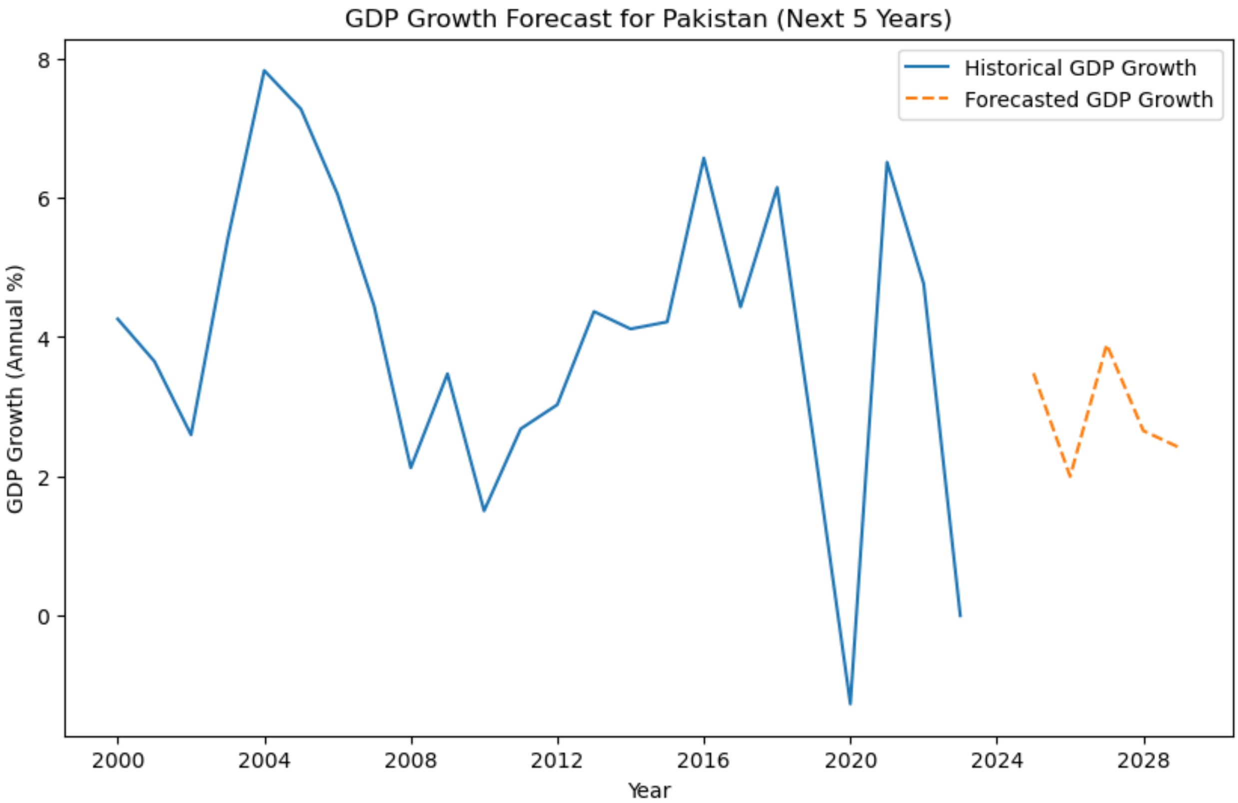

```
C:\Anaconda\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Value
Warning:

No frequency information was provided, so inferred frequency AS-JAN will b
e used.
```




```
C:\Anaconda\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Value
Warning:

No frequency information was provided, so inferred frequency AS-JAN will b
e used.
```



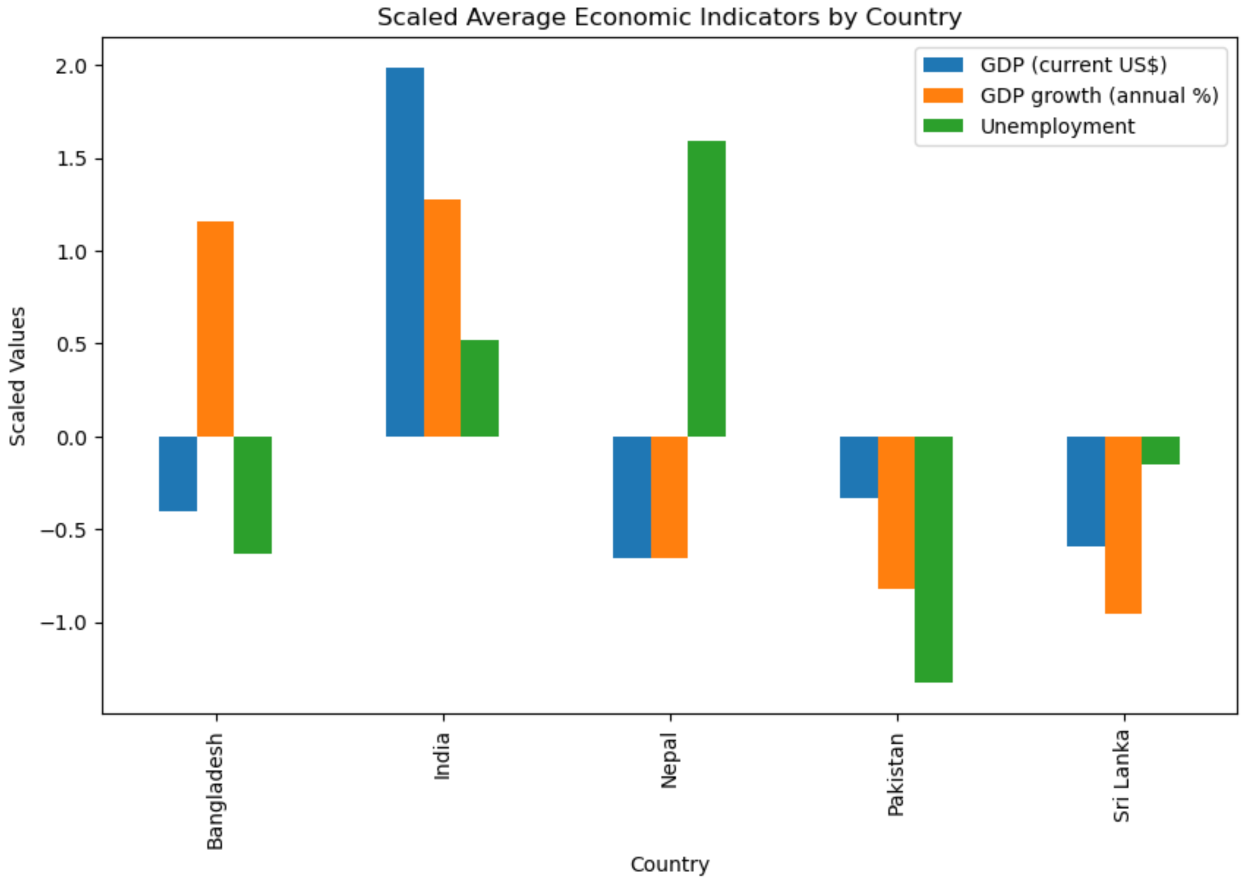
```
C:\Anaconda\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: Value
Warning:

No frequency information was provided, so inferred frequency AS-JAN will b
e used.
```



In [83]:

```python
from sklearn.preprocessing import StandardScaler
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px

# Calculate the average values by country
avg_economic = economic_growth_filtered.groupby("Country").mean().reset_i

# Select the columns to scale
economic_features = avg_economic[["Country", "GDP (current US$)", "GDP gr
                                  "Unemployment, total (% of total labor

# Scale only the numerical columns
scaler = StandardScaler()
scaled_features = scaler.fit_transform(economic_features[["GDP (current U
                                                          "GDP growth (an
                                                          "Unemployment,

# Convert scaled features back to a DataFrame with the country names
scaled_df = pd.DataFrame(scaled_features,
                         index=economic_features["Country"],
                         columns=["GDP (current US$)", "GDP growth (annua

# Plot the scaled data
scaled_df.plot(kind="bar", figsize=(10, 6))
```

```
plt.title("Scaled Average Economic Indicators by Country")
plt.ylabel("Scaled Values")
plt.show()

# Prepare the DataFrame for Plotly with cluster column if available
scaled_df.reset_index(inplace=True)
scaled_df["Cluster"] = economic_features.get("Cluster")  # Use .get to a

# Plotly scatter plot with hover info
fig = px.scatter(scaled_df,
                 x="GDP (current US$)",
                 y="Unemployment",
                 color="Cluster",
                 hover_name="Country",
                 title="KMeans Clustering of Economic Indicators by Coun

fig.show()
```




```
In [ ]:
```