# Solar Energy Potential Analysis

## Loading the dataset

In [15]:

```
# Install required packages if not installed
!pip install pandas numpy matplotlib seaborn scikit-learn openpyxl

# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
```

```
Requirement already satisfied: pandas in c:\users\reshm\anaconda3\lib\site-packages (2.2.3)
Requirement already satisfied: numpy in c:\users\reshm\anaconda3\lib\site-packages (1.26.4)
Requirement already satisfied: matplotlib in c:\users\reshm\anaconda3\lib\site-packages (3.7.1)
Requirement already satisfied: seaborn in c:\users\reshm\anaconda3\lib\site-packages (0.12.2)
Requirement already satisfied: scikit-learn in c:\users\reshm\anaconda3\lib\site-packages (1.5.2)
Requirement already satisfied: openpyxl in c:\users\reshm\anaconda3\lib\site-packages (3.1.5)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\reshm\anaconda3\lib\site-packages (from pa
ndas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\reshm\anaconda3\lib\site-packages (from pandas) (202
2.7)
Requirement already satisfied: tzdata>=2022.7 in c:\users\reshm\anaconda3\lib\site-packages (from pandas) (2
024.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\reshm\anaconda3\lib\site-packages (from matplotl
ib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\reshm\anaconda3\lib\site-packages (from matplotlib)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\reshm\anaconda3\lib\site-packages (from matplot
lib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\reshm\anaconda3\lib\site-packages (from matplot
lib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\reshm\anaconda3\lib\site-packages (from matplotli
b) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\reshm\anaconda3\lib\site-packages (from matplotlib)
(9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\reshm\anaconda3\lib\site-packages (from matplotl
ib) (3.0.9)
Requirement already satisfied: scipy>=1.6.0 in c:\users\reshm\anaconda3\lib\site-packages (from scikit-lear
n) (1.10.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\reshm\anaconda3\lib\site-packages (from scikit-lear
n) (1.2.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\reshm\anaconda3\lib\site-packages (from scik
it-learn) (3.5.0)
Requirement already satisfied: et-xmlfile in c:\users\reshm\anaconda3\lib\site-packages (from openpyxl) (1.
1.0)
Requirement already satisfied: six>=1.5 in c:\users\reshm\anaconda3\lib\site-packages (from python-dateutil>
=2.8.2->pandas) (1.16.0)
```

In [28]:

```python
# Load the dataset
file_path = "D:/Capstone_2025/data/solardata_addis.xlsx"  # Update with your actual path
df = pd.read_excel(file_path, sheet_name='Sheet1')
df
```

Out[28]:

| | Year | Month | Day | Hour | Minute | Clearsky DHI | Clearsky DNI | Temperature | Clearsky GHI | cloud fill flag | ... | DNI | Fill Flag | GHI | Relative Humidity | Solar Zenith Angle | Surfa Albe |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2006 | 1 | 1 | 0 | 30 | 0 | 0 | 6.9 | 0 | 0 | ... | 0 | 0 | 0 | 80.78 | 137.73 | 0 |
| **1** | 2006 | 1 | 1 | 1 | 30 | 0 | 0 | 6.5 | 0 | 0 | ... | 0 | 0 | 0 | 85.35 | 124.08 | 0 |
| **2** | 2006 | 1 | 1 | 2 | 30 | 0 | 0 | 6.2 | 0 | 0 | ... | 0 | 0 | 0 | 89.38 | 110.30 | 0 |
| **3** | 2006 | 1 | 1 | 3 | 30 | 0 | 0 | 7.0 | 0 | 0 | ... | 0 | 0 | 0 | 86.64 | 96.54 | 0 |
| **4** | 2006 | 1 | 1 | 4 | 30 | 36 | 484 | 10.8 | 96 | 1 | ... | 0 | 1 | 39 | 68.50 | 82.85 | 0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **17515** | 2022 | 1 | 31 | 19 | 30 | 0 | 0 | 13.7 | 0 | 0 | ... | 0 | 0 | 0 | 64.84 | 144.22 | 0 |
| **17516** | 2022 | 1 | 31 | 20 | 30 | 0 | 0 | 13.6 | 0 | 0 | ... | 0 | 0 | 0 | 63.55 | 158.14 | 0 |
| **17517** | 2022 | 1 | 31 | 21 | 30 | 0 | 0 | 13.3 | 0 | 0 | ... | 0 | 0 | 0 | 64.07 | 170.03 | 0 |
| **17518** | 2022 | 1 | 31 | 22 | 30 | 0 | 0 | 12.8 | 0 | 0 | ... | 0 | 0 | 0 | 63.95 | 167.90 | 0 |
| **17519** | 2022 | 1 | 31 | 23 | 30 | 0 | 0 | 12.1 | 0 | 0 | ... | 0 | 0 | 0 | 65.48 | 155.12 | 0 |

17520 rows × 23 columns

In [29]:

```
# Display dataset structure
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17520 entries, 0 to 17519
Data columns (total 23 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Year                17520 non-null  int64  
 1   Month               17520 non-null  int64  
 2   Day                 17520 non-null  int64  
 3   Hour                17520 non-null  int64  
 4   Minute              17520 non-null  int64  
 5   Clearsky DHI        17520 non-null  int64  
 6   Clearsky DNI        17520 non-null  int64  
 7   Temperature         17520 non-null  float64
 8   Clearsky GHI        17520 non-null  int64  
 9   cloud fill flag     17520 non-null  int64  
 10  Cloud Type          17520 non-null  int64  
 11  Dew Point           17520 non-null  float64
 12  DHI                 17520 non-null  int64  
 13  DNI                 17520 non-null  int64  
 14  Fill Flag           17520 non-null  int64  
 15  GHI                 17520 non-null  int64  
 16  Relative Humidity   17520 non-null  float64
 17  Solar Zenith Angle  17520 non-null  float64
 18  Surface Albedo      17520 non-null  float64
 19  Pressure            17520 non-null  int64  
 20  Precipitable Water  17520 non-null  float64
 21  Wind Direction      17520 non-null  int64  
 22  Wind Speed          17520 non-null  float64
dtypes: float64(7), int64(16)
memory usage: 3.1 MB
None
   Year  Month  Day  Hour  Minute  Clearsky DHI  Clearsky DNI  Temperature  \
0  2006      1    1     0      30             0             0          6.9   
1  2006      1    1     1      30             0             0          6.5   
2  2006      1    1     2      30             0             0          6.2   
3  2006      1    1     3      30             0             0          7.0   
4  2006      1    1     4      30            36           484         10.8   

   Clearsky GHI  cloud fill flag  ...  DNI  Fill Flag  GHI  Relative Humidity  \
0             0                0  ...    0          0    0              80.78   
1             0                0  ...    0          0    0              85.35   
2             0                0  ...    0          0    0              89.38   
3             0                0  ...    0          0    0              86.64   
```

```
4           96                1  ...    0          1   39              68.50

   Solar Zenith Angle  Surface Albedo  Pressure  Precipitable Water  \
0              137.73            0.15       778                 0.9
1              124.08            0.15       779                 1.0
2              110.30            0.15       779                 1.0
3               96.54            0.15       780                 1.0
4               82.85            0.15       780                 1.0

   Wind Direction  Wind Speed
0              85         1.7
1              89         1.7
2              92         1.7
3              94         2.1
4              97         3.5

[5 rows x 23 columns]
```

In [30]:
```
#handling missing values
# Check for missing values
print("Missing Values:\n", df.isnull().sum())
```

```
Missing Values:
 Year                 0
Month                0
Day                  0
Hour                 0
Minute               0
Clearsky DHI         0
Clearsky DNI         0
Temperature          0
Clearsky GHI         0
cloud fill flag      0
Cloud Type           0
Dew Point            0
DHI                  0
DNI                  0
Fill Flag            0
GHI                  0
Relative Humidity    0
Solar Zenith Angle   0
Surface Albedo       0
Pressure             0
Precipitable Water   0
Wind Direction       0
Wind Speed           0
dtype: int64
```

In [31]:
```
# Fill missing values using forward and backward fill
df.fillna(method='ffill', inplace=True)
df.fillna(method='bfill', inplace=True)
```

```
C:\Users\reshm\AppData\Local\Temp\ipykernel_27192\212673031.py:2: FutureWarning: DataFrame.fillna with 'meth
od' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method='ffill', inplace=True)
C:\Users\reshm\AppData\Local\Temp\ipykernel_27192\212673031.py:3: FutureWarning: DataFrame.fillna with 'meth
od' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  df.fillna(method='bfill', inplace=True)
```

In [32]:
```
# Verify no missing values remain
print("Missing Values after filling:\n", df.isnull().sum())
```

```
Missing Values after filling:
 Year                  0
Month                 0
Day                   0
Hour                  0
Minute                0
Clearsky DHI          0
Clearsky DNI          0
Temperature           0
Clearsky GHI          0
cloud fill flag       0
Cloud Type            0
Dew Point             0
DHI                   0
DNI                   0
Fill Flag             0
GHI                   0
Relative Humidity     0
Solar Zenith Angle    0
Surface Albedo        0
Pressure              0
Precipitable Water    0
Wind Direction        0
Wind Speed            0
dtype: int64
```

In [33]:

```
#create datetime features
# Convert date-related columns into a Datetime object
df['Datetime'] = pd.to_datetime(df[['Year', 'Month', 'Day', 'Hour']])

# Set Datetime as the index
df.set_index('Datetime', inplace=True)

# Drop redundant columns
df.drop(columns=['Minute'], inplace=True, errors='ignore')

# Extract additional time-based features
df['DayOfYear'] = df.index.dayofyear
df['WeekOfYear'] = df.index.isocalendar().week
df['Month'] = df.index.month
df['Hour'] = df.index.hour
df['Season'] = df.index.month.map(lambda m: 'Winter' if m in [12, 1, 2] else
                                  'Spring' if m in [3, 4, 5] else
                                  'Summer' if m in [6, 7, 8] else 'Autumn')
```

In [34]:
```
# Display updated dataset
df.head()
```

Out[34]:

| Datetime | Year | Month | Day | Hour | Clearsky DHI | Clearsky DNI | Temperature | Clearsky GHI | cloud fill flag | Cloud Type | ... | Relative Humidity | Solar Zenith Angle | Surface Albedo | Pressure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2006-01-01 00:00:00 | 2006 | 1 | 1 | 0 | 0 | 0 | 6.9 | 0 | 0 | 0 | ... | 80.78 | 137.73 | 0.15 | 778 |
| 2006-01-01 01:00:00 | 2006 | 1 | 1 | 1 | 0 | 0 | 6.5 | 0 | 0 | 0 | ... | 85.35 | 124.08 | 0.15 | 779 |
| 2006-01-01 02:00:00 | 2006 | 1 | 1 | 2 | 0 | 0 | 6.2 | 0 | 0 | 0 | ... | 89.38 | 110.30 | 0.15 | 779 |
| 2006-01-01 03:00:00 | 2006 | 1 | 1 | 3 | 0 | 0 | 7.0 | 0 | 0 | 0 | ... | 86.64 | 96.54 | 0.15 | 780 |
| 2006-01-01 04:00:00 | 2006 | 1 | 1 | 4 | 36 | 484 | 10.8 | 96 | 1 | 3 | ... | 68.50 | 82.85 | 0.15 | 780 |

5 rows × 25 columns

## Feature engineering

In [35]:
```
#Estimate Solar Power Output (Simplified)
# Assuming 20% efficiency of solar panels
df['Estimated_Solar_Power_kW'] = df['GHI'] * 0.2 / 1000  # Convert W/m² to kW/m²
```

In [36]:
```
# Adjust Wind Speed for Power Estimation
# Wind Power Estimation using P = 0.5 * air_density * A * v³
air_density = 1.225  # kg/m³ (sea level standard)
turbine_swept_area = 50  # Assumed 50m² (adjust as needed)

df['Estimated_Wind_Power_kW'] = 0.5 * air_density * turbine_swept_area * (df['Wind Speed'] ** 3) / 1000
```

## Outlier detection and removal

In [37]:
```
# Function to remove outliers using IQR method
def remove_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] >= lower_bound) & (df[column] <= upper_bound)]
```

In [39]:
```python
# Apply to key columns
df = remove_outliers(df, 'GHI')
df = remove_outliers(df, 'DNI')
df = remove_outliers(df, 'Wind Speed')

# Display updated dataset
df.describe()
```

Out[39]:

| | Year | Month | Day | Hour | Clearsky DHI | Clearsky DNI | Temperature | Clearsky GHI | cloud fill flag |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 13436.000000 | 13436.000000 | 13436.000000 | 13436.000000 | 13436.000000 | 13436.000000 | 13436.000000 | 13436.000000 | 13436.000000 |
| **mean** | 2012.765630 | 6.624367 | 15.694850 | 12.288255 | 41.885457 | 205.874963 | 14.919024 | 180.394686 | 0.582167 |
| **std** | 5.696955 | 3.247457 | 8.800264 | 7.568925 | 71.409216 | 328.229953 | 4.384039 | 324.124282 | 1.934050 |
| **min** | 2005.000000 | 1.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 4.000000 | 0.000000 | 0.000000 |
| **25%** | 2007.000000 | 4.000000 | 8.000000 | 4.000000 | 0.000000 | 0.000000 | 12.100000 | 0.000000 | 0.000000 |
| **50%** | 2013.000000 | 7.000000 | 16.000000 | 14.000000 | 0.000000 | 0.000000 | 14.400000 | 0.000000 | 0.000000 |
| **75%** | 2018.000000 | 9.000000 | 23.000000 | 19.000000 | 76.000000 | 447.000000 | 17.800000 | 197.000000 | 0.000000 |
| **max** | 2022.000000 | 12.000000 | 31.000000 | 23.000000 | 452.000000 | 1032.000000 | 30.700000 | 1114.000000 | 8.000000 |

8 rows × 26 columns

In [40]:
```
#Normalise the data
scaler = MinMaxScaler()
features = ['GHI', 'DNI', 'DHI', 'Temperature', 'Relative Humidity', 'Wind Speed', 'Estimated_Solar_Power_kW'

df[features] = scaler.fit_transform(df[features])

# Display final dataset
df.head()
```

Out[40]:

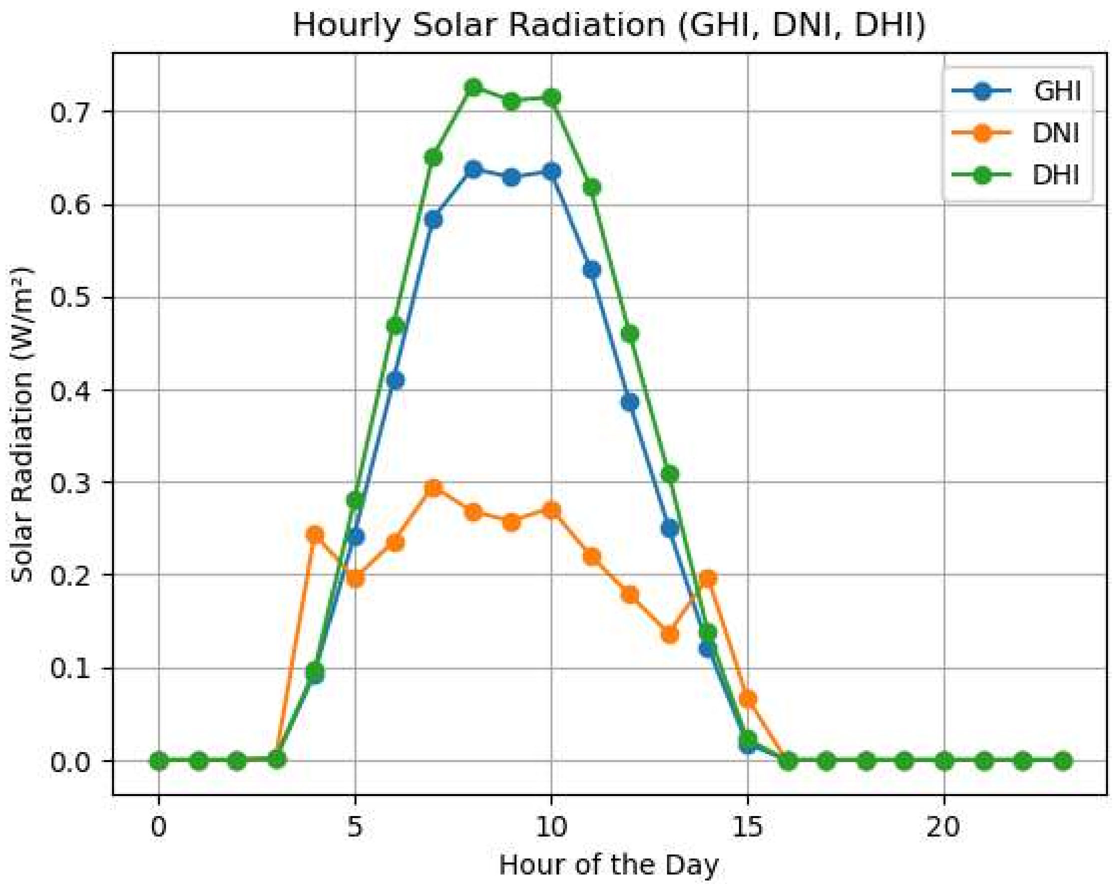
| | Year | Month | Day | Hour | Clearsky DHI | Clearsky DNI | Temperature | Clearsky GHI | cloud fill flag | Cloud Type | ... | Surface Albedo | Pressure | Precipitable Water | W Direc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Datetime** | | | | | | | | | | | | | | | |
| **2006-01-01 00:00:00** | 2006 | 1 | 1 | 0 | 0 | 0 | 0.108614 | 0 | 0 | 0 | ... | 0.15 | 778 | 0.9 | |
| **2006-01-01 01:00:00** | 2006 | 1 | 1 | 1 | 0 | 0 | 0.093633 | 0 | 0 | 0 | ... | 0.15 | 779 | 1.0 | |
| **2006-01-01 02:00:00** | 2006 | 1 | 1 | 2 | 0 | 0 | 0.082397 | 0 | 0 | 0 | ... | 0.15 | 779 | 1.0 | |
| **2006-01-01 03:00:00** | 2006 | 1 | 1 | 3 | 0 | 0 | 0.112360 | 0 | 0 | 0 | ... | 0.15 | 780 | 1.0 | |
| **2006-01-01 04:00:00** | 2006 | 1 | 1 | 4 | 36 | 484 | 0.254682 | 96 | 1 | 3 | ... | 0.15 | 780 | 1.0 | |

5 rows × 27 columns

### Exploratory Data Analysis (EDA)

In [41]:
```
#Visualizing Hourly Solar Energy Trends
# Aggregate hourly mean values
hourly_avg = df.groupby(df.index.hour)[['GHI', 'DNI', 'DHI']].mean()

# Plot hourly variations
plt.figure(figsize=(12, 5))
hourly_avg.plot(marker='o', linestyle='-')
plt.title("Hourly Solar Radiation (GHI, DNI, DHI)")
plt.xlabel("Hour of the Day")
plt.ylabel("Solar Radiation (W/m²)")
plt.grid(True)
plt.legend()
plt.show()
```

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

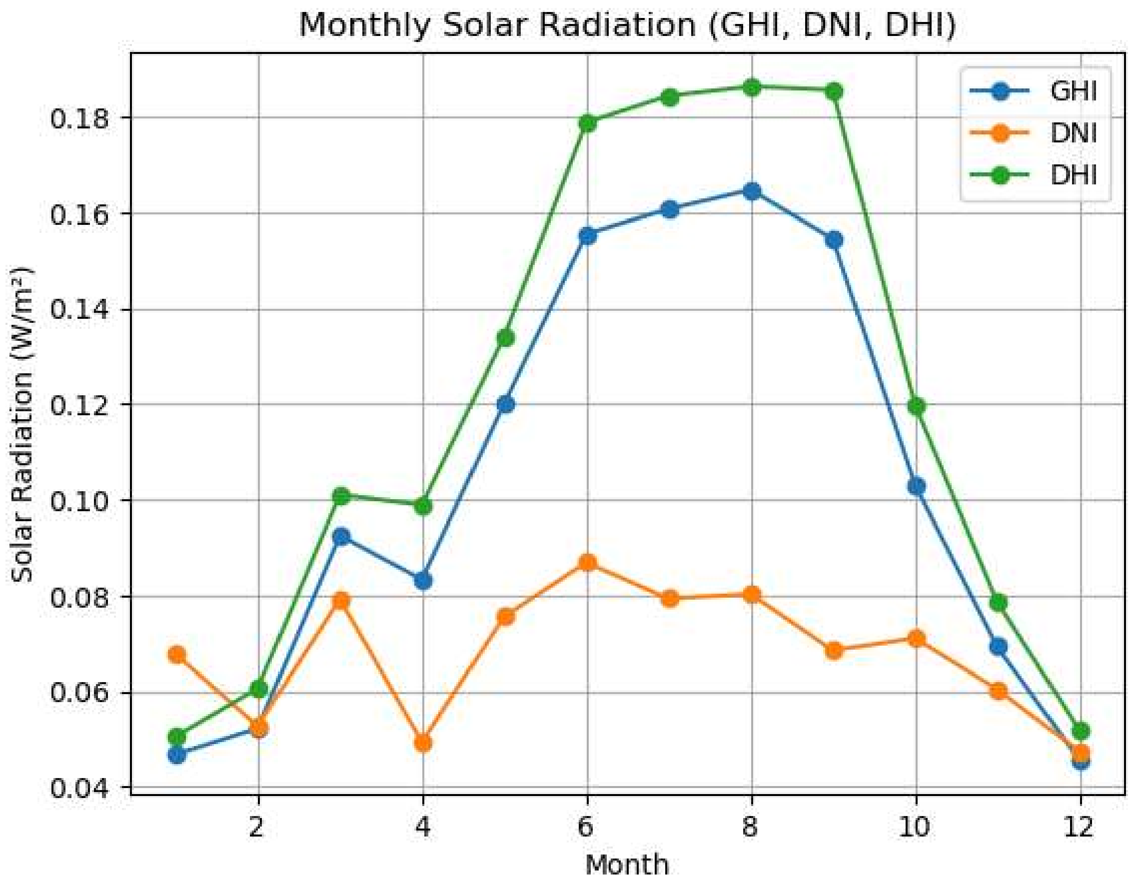


Hourly Solar Radiation (GHI, DNI, DHI)
GHI
DNI
DHI
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Solar Radiation (W/m²)
0
5
10
15
20
Hour of the Day

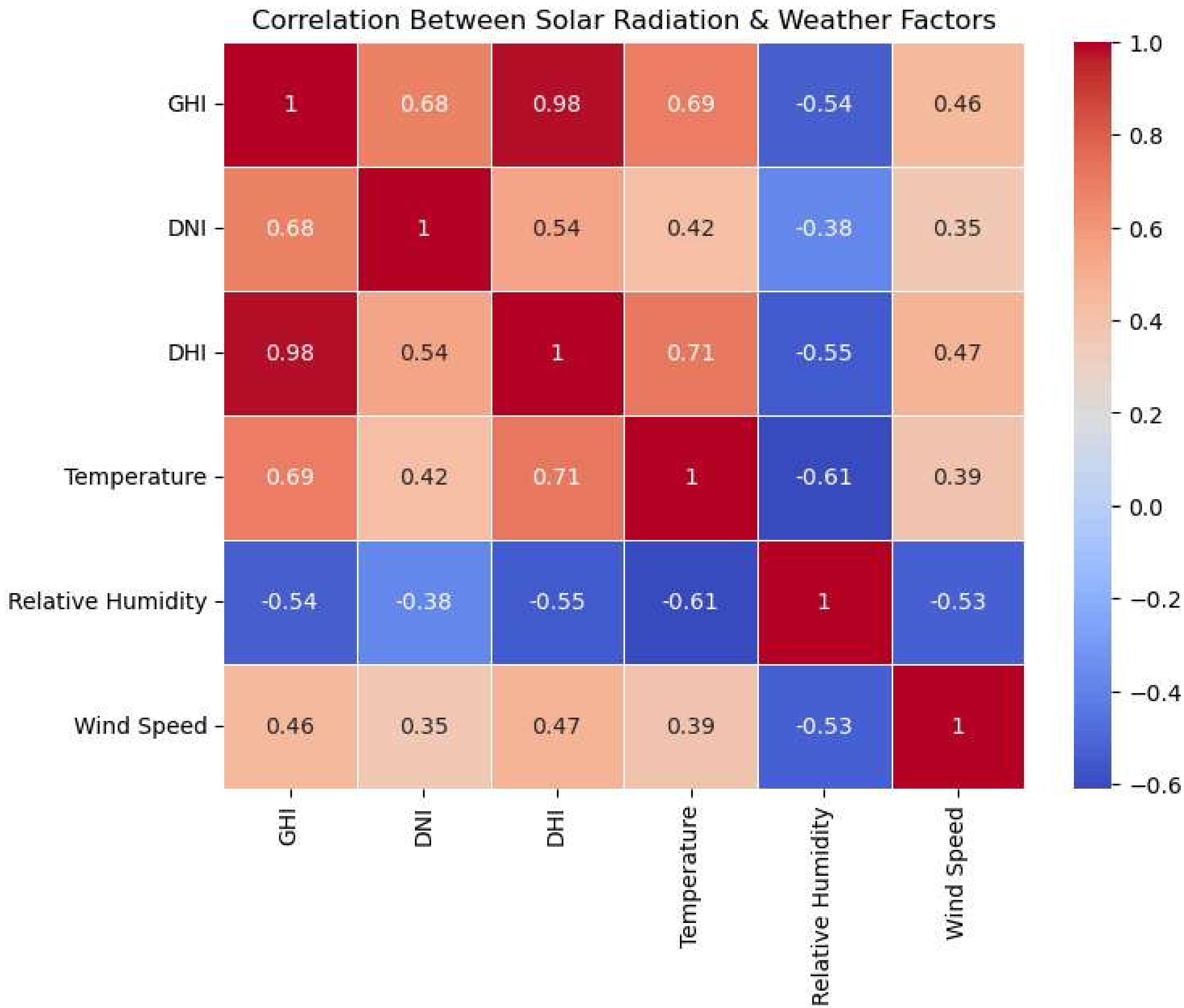
In [42]:

```python
# Monthly Solar Energy Potential
# Aggregate monthly mean values
monthly_avg = df.groupby(df.index.month)[['GHI', 'DNI', 'DHI']].mean()

# Plot monthly variations
plt.figure(figsize=(12, 5))
monthly_avg.plot(marker='o', linestyle='-')
plt.title("Monthly Solar Radiation (GHI, DNI, DHI)")
plt.xlabel("Month")
plt.ylabel("Solar Radiation (W/m²)")
plt.grid(True)
plt.legend()
plt.show()
```

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

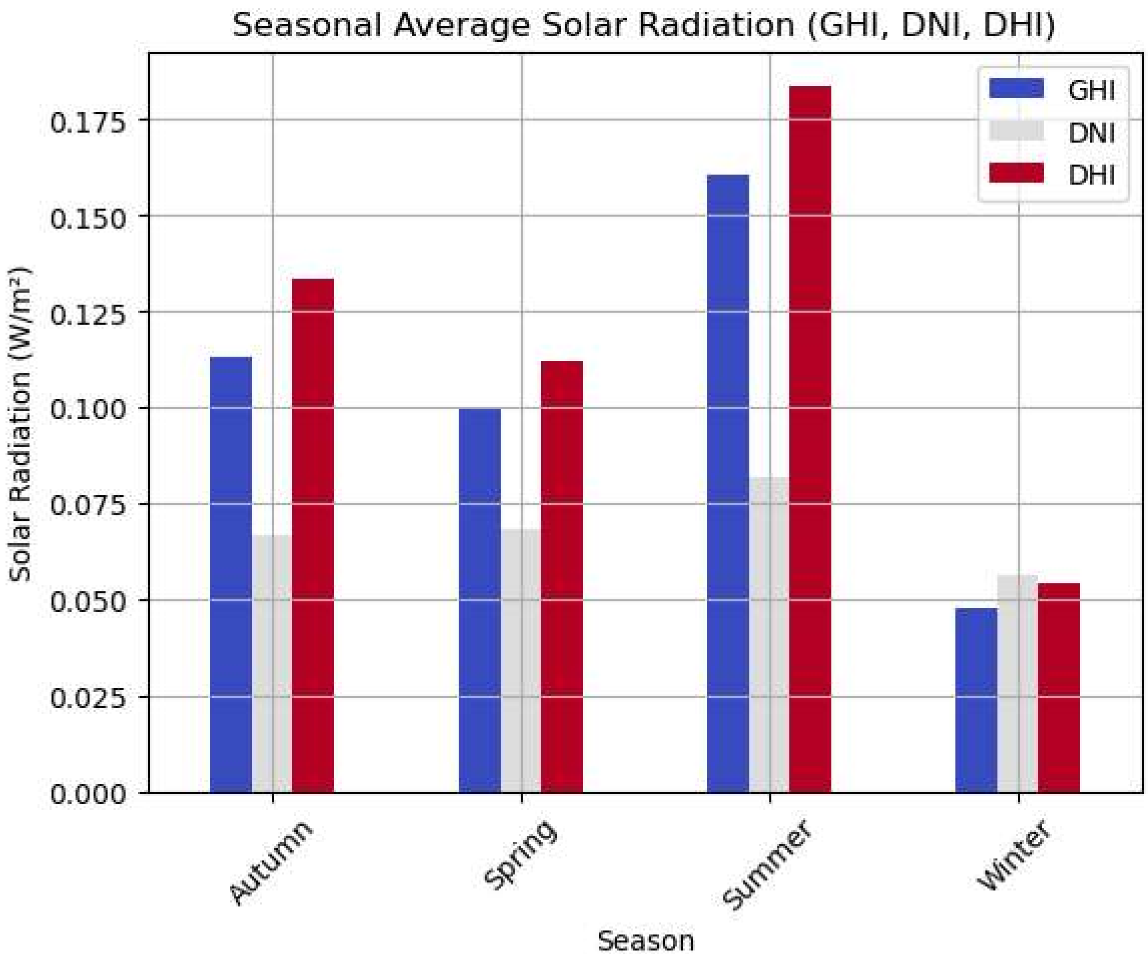

Monthly Solar Radiation (GHI, DNI, DHI)
GHI
DNI
DHI
Solar Radiation (W/m²)
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
2
4
6
8
10
12
Month

In [43]:
```
#Correlation Analysis
# Compute correlation matrix
correlation_matrix = df[['GHI', 'DNI', 'DHI', 'Temperature', 'Relative Humidity', 'Wind Speed']].corr()

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Between Solar Radiation & Weather Factors")
plt.show()
```


Correlation Between Solar Radiation & Weather Factors
GHI
DNI
DHI
Temperature
Relative Humidity
Wind Speed
1 0.68 0.98 0.69 -0.54 0.46
0.68 1 0.54 0.42 -0.38 0.35
0.98 0.54 1 0.71 -0.55 0.47
0.69 0.42 0.71 1 -0.61 0.39
-0.54 -0.38 -0.55 -0.61 1 -0.53
0.46 0.35 0.47 0.39 -0.53 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

In [44]:
```
# Identify Seasonal Variations
# Define seasons based on the Northern Hemisphere
df['Season'] = df.index.month.map(lambda m: 'Winter' if m in [12, 1, 2] else
                                  'Spring' if m in [3, 4, 5] else
                                  'Summer' if m in [6, 7, 8] else 'Autumn')

# Compute seasonal average values
seasonal_avg = df.groupby('Season')[['GHI', 'DNI', 'DHI']].mean()

# Plot seasonal variations
plt.figure(figsize=(8, 5))
seasonal_avg.plot(kind='bar', colormap='coolwarm')
plt.title("Seasonal Average Solar Radiation (GHI, DNI, DHI)")
plt.xlabel("Season")
plt.ylabel("Solar Radiation (W/m²)")
plt.xticks(rotation=45)
plt.grid(True)
plt.legend()
plt.show()
```

```
<Figure size 800x500 with 0 Axes>
```




In [45]:
```
# Save as CSV
csv_path = 'preprocessed_data.csv'
df.to_csv(csv_path,index=False)
print(f"Preprocessed data saved as CSV: {csv_path}")
```

Preprocessed data saved as CSV: preprocessed_data.csv

In [ ]: