Import Libaries and Load Data

In [7]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense
from tensorflow.keras.optimizers import Adam

# Read the CSV file into a pandas DataFrame
file_path = "london_weather[1].csv"
df = pd.read_csv(file_path)

# Prepare the data
df['date'] = pd.to_datetime(df['date'], format='%Y%m%d')
df.set_index('date', inplace=True)
mean_temp_data = df['mean_temp'].dropna()
```

Training and Testing

In [8]:
```
# Normalize the data
scaler = MinMaxScaler()
normalized_data = scaler.fit_transform(mean_temp_data.values.reshape(-1, 1))

# Split data into train and test sets
train_size = int(len(normalized_data) * 0.8)
train_data, test_data = normalized_data[:train_size], normalized_data[train_size:]
```

Model Building

In [9]:
```
# Create sequences for training
sequence_length = 30
X_train, y_train = [], []
for i in range(len(train_data) - sequence_length):
    X_train.append(train_data[i:i+sequence_length])
    y_train.append(train_data[i+sequence_length])
X_train, y_train = np.array(X_train), np.array(y_train)

# Build the GRU model
model = Sequential([
    GRU(units=50, activation='relu', input_shape=(sequence_length, 1)),
    Dense(units=1)
])
model.compile(optimizer=Adam(learning_rate=0.001), loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=1)

# Make predictions on test data
X_test = []
for i in range(len(test_data) - sequence_length):
    X_test.append(test_data[i:i+sequence_length])
X_test = np.array(X_test)
y_pred = model.predict(X_test)
```

```python
# Inverse transform to get the original scale
y_pred = scaler.inverse_transform(y_pred)
test_data = scaler.inverse_transform(test_data[sequence_length:])

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(test_data, y_pred))
print("Root Mean Squared Error:", rmse)
```

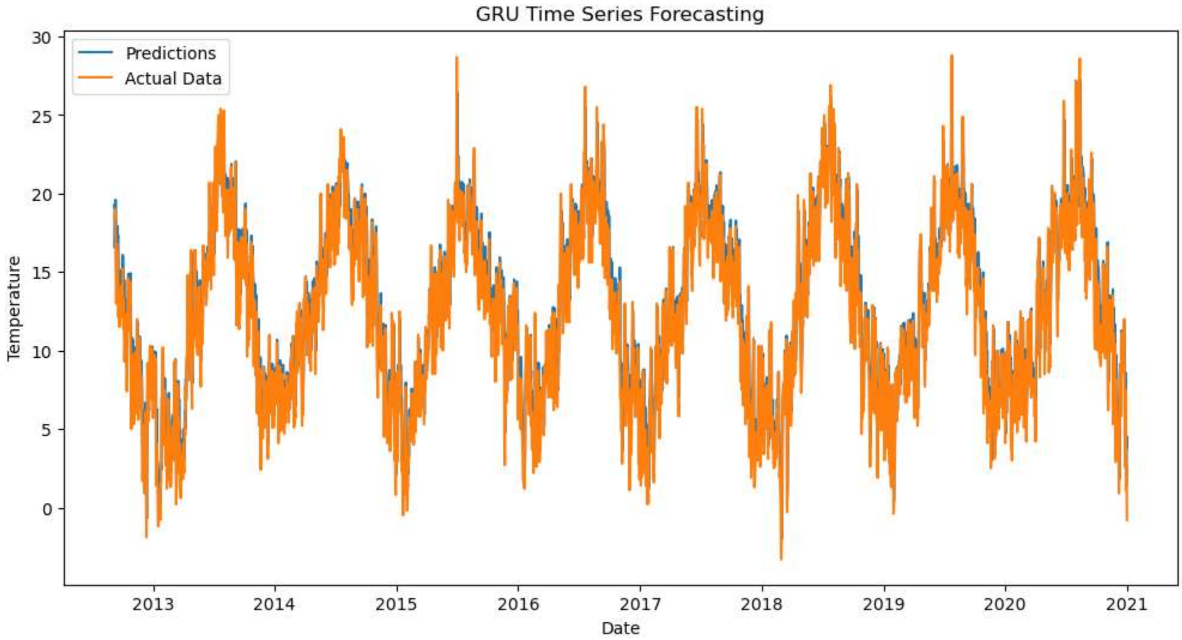
```
Epoch 1/50
382/382 [==============================] - 6s 11ms/step - loss: 0.0135
Epoch 2/50
382/382 [==============================] - 4s 11ms/step - loss: 0.0030
Epoch 3/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0028
Epoch 4/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 5/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 6/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 7/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0027
Epoch 8/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 9/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0027
Epoch 10/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0027
Epoch 11/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 12/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 13/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 14/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 15/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 16/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 17/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 18/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 19/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 20/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0026
Epoch 21/50
382/382 [==============================] - 5s 14ms/step - loss: 0.0027
Epoch 22/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0027
Epoch 23/50
382/382 [==============================] - 6s 15ms/step - loss: 0.0027
Epoch 24/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0027
Epoch 25/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0027
Epoch 26/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0026
Epoch 27/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0026
Epoch 28/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 29/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 30/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 31/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0026
Epoch 32/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
```

```
Epoch 33/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 34/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 35/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 36/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 37/50
382/382 [==============================] - 5s 14ms/step - loss: 0.0026
Epoch 38/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 39/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0026
Epoch 40/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 41/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0027
Epoch 42/50
382/382 [==============================] - 5s 12ms/step - loss: 0.0026
Epoch 43/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 44/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 45/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0027
Epoch 46/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 47/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 48/50
382/382 [==============================] - 5s 14ms/step - loss: 0.0026
Epoch 49/50
382/382 [==============================] - 5s 13ms/step - loss: 0.0026
Epoch 50/50
382/382 [==============================] - 6s 15ms/step - loss: 0.0027
95/95 [==============================] - 1s 4ms/step
Root Mean Squared Error: 2.0985093277711364
```

Plot Results

In [10]:

```
# Plot the results
plt.figure(figsize=(12, 6))
plt.plot(mean_temp_data.index[train_size+sequence_length:], y_pred, label='Predicti
plt.plot(mean_temp_data.index[train_size+sequence_length:], test_data, label='Actua
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.title('GRU Time Series Forecasting')
plt.legend()
plt.show()
```


GRU Time Series Forecasting
Predictions
Actual Data
Temperature
Date
30
25
20
15
10
5
0
2013
2014
2015
2016
2017
2018
2019
2020
2021

```
In [ ]:
```