In [2]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-pytho
n
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files
under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserve
d as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of
the current session
```

```
/kaggle/input/treated-data-irr/Final treated data (1).xlsx
/kaggle/input/treated-data-irrigation/Final treated data.xlsx
```

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import SimpleRNN
from sklearn.metrics import mean_absolute_error, mean_squared_error
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
2024-05-02 08:40:50.747040: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:926
1] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when
one has already been registered
2024-05-02 08:40:50.747192: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607
] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when
one has already been registered
2024-05-02 08:40:50.906530: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:15
15] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS w
hen one has already been registered
```

In [4]:

```
df = pd.read_excel('/kaggle/input/treated-data-irr/Final treated data (1).xlsx')
```

In [5]:

```
df.head()
```

Out[5]:

	Date	pH	Temp	TDS	TSS	Cl	BOD	COD	WWQIs	WWQIi
0	2021-07-02	7.6	30.0	316.0	18.0	76.0	1.5	33.6	16.83	13.65

	Date	pH	Temp	TDS	TSS	Cl	BOD	COD	WWQIs	WWQIi
1	2021-07-05	7.2	29.0	240.0	36.0	43.0	1.5	8.4	12.15	9.10
2	2021-07-07	7.3	30.0	276.0	21.0	51.0	0.8	16.8	12.82	9.28
3	2021-07-09	7.1	28.0	194.0	15.0	32.0	1.4	8.4	7.47	5.39
4	2021-07-12	7.5	29.0	198.0	16.0	36.0	1.1	8.4	11.51	10.00

In [6]:

```
dfi = df.copy()
```

In [7]:

```
dfs = dfi.drop('WWQIs', axis=1)
```

In [8]:

```
for column in dfs.columns.difference(['Date']):
    mean_value = dfs[column].mean()
    dfs[column].fillna(mean_value, inplace=True)
```

```
/tmp/ipykernel_33/4021660727.py:3: FutureWarning: A value is trying to be set on a copy o
f a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the i
ntermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col:
value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operati
on inplace on the original object.


  dfs[column].fillna(mean_value, inplace=True)
```

In [9]:

```
null_percentage = (dfs.isnull().sum() / len(dfs)) * 100
null_percentage
```

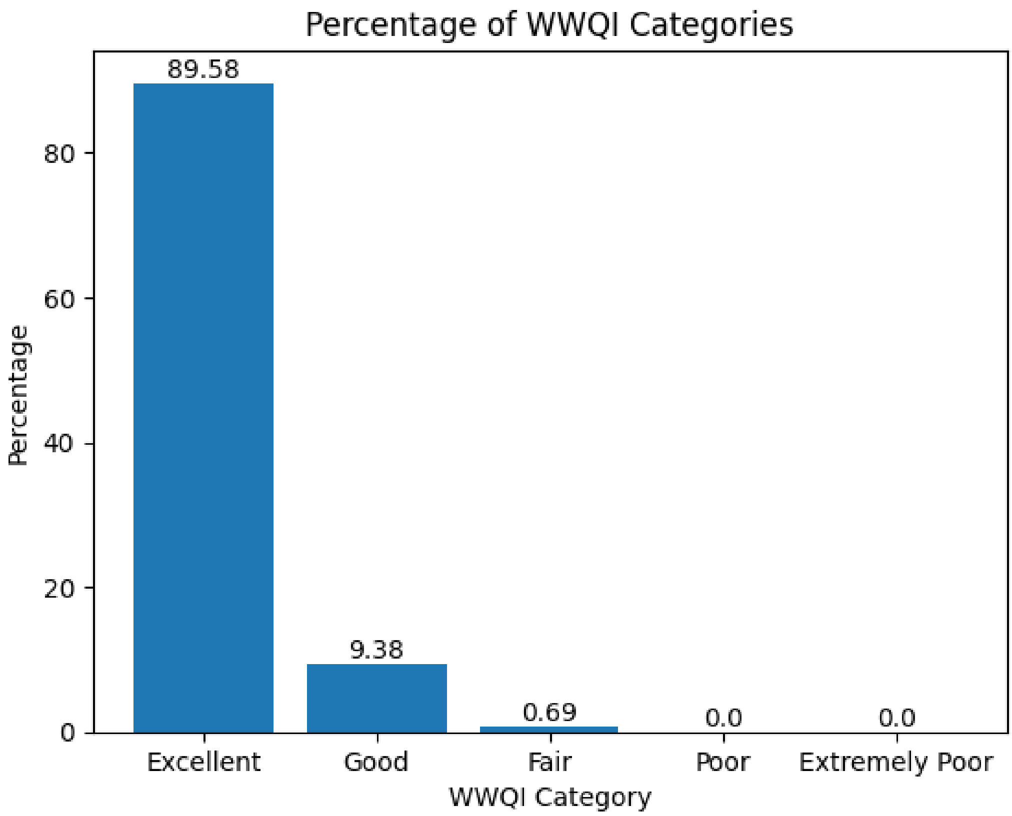
Out[9]:

```
Date     0.0
pH       0.0
Temp     0.0
TDS      0.0
TSS      0.0
Cl       0.0
BOD      0.0
COD      0.0
WWQIi    0.0
dtype: float64
```

In [10]:

```
# Calculate WWQI categories
conditions = [
    (dfs['WWQIi'] < 25),
    (dfs['WWQIi'] >= 25) & (df['WWQIi'] < 50),
    (dfs['WWQIi'] >= 50) & (df['WWQIi'] < 75),
    (dfs['WWQIi'] >= 75) & (df['WWQIi'] < 100),
    (dfs['WWQIi'] >= 100)
]
categories = ['Excellent', 'Good', 'Fair', 'Poor', 'Extremely Poor']

# Create a new column 'WWQI_Category'
dfs['WWQI_Category'] = pd.cut(df['WWQIi'], bins=[-float('inf'), 25, 50, 75, 100, float('
inf')],
                              labels=categories)

# Count occurrences of each category
category_counts = dfs['WWQI_Category'].value_counts()
```

```python
# Calculate percentages
category_percentages = (category_counts / len(dfs)) * 100

fig, ax = plt.subplots()
bars = ax.bar(category_percentages.index, category_percentages.values)

# Display numbers above the bars
for bar in bars:
    yval = bar.get_height()
    ax.text(bar.get_x() + bar.get_width() / 2, yval, round(yval, 2), ha='center', va='bo
ttom')

plt.xlabel('WWQI Category')
plt.ylabel('Percentage')
plt.title('Percentage of WWQI Categories')
plt.show()
```



In [11]:

```python
dfs.drop("WWQI_Category", axis=1, inplace=True)
```

In [12]:

```python
dfs['Date'] = pd.to_datetime(dfs['Date'])
dfs.set_index('Date', inplace=True)
```

In [13]:

```python
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(dfs)
```

In [14]:

```python
def create_sequences(data, seq_length):
    sequences = []
    targets = []

    for i in range(len(data) - seq_length):
        seq = data[i:i + seq_length]
        label = data[i + seq_length]
        sequences.append(seq)
        targets.append(label)
```

```
    return np.array(sequences), np.array(targets)

seq_length = 10  # You can adjust this based on your needs
X, y = create_sequences(scaled_data, seq_length)
```

In [15]:

```

```

In [15]:

```
def custom_train_test_split(X, y, test_size=0.2, random_state=None):
    classes = np.unique(y)
    train_indices, test_indices = [], []
    for c in classes:
        indices = np.where(y == c)[0]
        np.random.shuffle(indices)
        split_idx = int(len(indices) * (1 - test_size))
        train_indices.extend(indices[:split_idx])
        test_indices.extend(indices[split_idx:])
    np.random.shuffle(train_indices)
    np.random.shuffle(test_indices)
    X_train, X_test = X[train_indices], X[test_indices]
    y_train, y_test = y[train_indices], y[test_indices]
    return X_train, X_test, y_train, y_test

X_train, X_test, y_train, y_test = custom_train_test_split(X, y, test_size=0.2, random_s
tate=42)
```

In [17]:

```
X_train.shape
```

Out[17]:

```
(1234, 10, 8)
```

In [16]:

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True
)
```

RNN Modelling

In [17]:

```
modelRnn = Sequential()
modelRnn.add(SimpleRNN(units=16, return_sequences=True, input_shape=(X_train.shape[1], X
_train.shape[2])))
modelRnn.add(SimpleRNN(units=64))
modelRnn.add(Dense(units=y_train.shape[1]))  # Assuming the output size is the same as in
put size

modelRnn.compile(optimizer='adam', loss='mean_squared_error')

historyRnn = modelRnn.fit(X_train, y_train, epochs=200, batch_size=24, validation_data=(
X_test, y_test), verbose=1, callbacks=[early_stopping])
```

```
Epoch 1/200
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/layers/rnn/rnn.py:204: UserWarning: Do
not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

```
52/52 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - loss: 0.1694 - val_loss: 0.0336
Epoch 2/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0266 - val_loss: 0.0281
Epoch 3/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0246 - val_loss: 0.0260
```

```
Epoch 4/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0220 - val_loss: 0.0246
Epoch 5/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0203 - val_loss: 0.0257
Epoch 6/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0208 - val_loss: 0.0245
Epoch 7/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0214 - val_loss: 0.0230
Epoch 8/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0206 - val_loss: 0.0227
Epoch 9/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0191 - val_loss: 0.0236
Epoch 10/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0194 - val_loss: 0.0227
Epoch 11/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0190 - val_loss: 0.0220
Epoch 12/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0187 - val_loss: 0.0222
Epoch 13/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0189 - val_loss: 0.0223
Epoch 14/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0193 - val_loss: 0.0216
Epoch 15/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0178 - val_loss: 0.0213
Epoch 16/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0175 - val_loss: 0.0207
Epoch 17/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0174 - val_loss: 0.0204
Epoch 18/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0183 - val_loss: 0.0209
Epoch 19/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0185 - val_loss: 0.0206
Epoch 20/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0171 - val_loss: 0.0210
Epoch 21/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0173 - val_loss: 0.0205
Epoch 22/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0173 - val_loss: 0.0200
Epoch 23/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0169 - val_loss: 0.0190
Epoch 24/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0165 - val_loss: 0.0206
Epoch 25/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0168 - val_loss: 0.0188
Epoch 26/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0162 - val_loss: 0.0190
Epoch 27/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0154 - val_loss: 0.0186
Epoch 28/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0162 - val_loss: 0.0184
Epoch 29/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0153 - val_loss: 0.0180
Epoch 30/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0142 - val_loss: 0.0178
Epoch 31/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0150 - val_loss: 0.0170
Epoch 32/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0147 - val_loss: 0.0178
Epoch 33/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0144 - val_loss: 0.0171
Epoch 34/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0140 - val_loss: 0.0166
Epoch 35/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0135 - val_loss: 0.0160
Epoch 36/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0133 - val_loss: 0.0160
Epoch 37/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0133 - val_loss: 0.0151
Epoch 38/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0125 - val_loss: 0.0148
Epoch 39/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0123 - val_loss: 0.0148
Epoch 40/200
```

```
Epoch 40/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0121 - val_loss: 0.0154
Epoch 41/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0117 - val_loss: 0.0141
Epoch 42/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0118 - val_loss: 0.0138
Epoch 43/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0108 - val_loss: 0.0143
Epoch 44/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0116 - val_loss: 0.0130
Epoch 45/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0107 - val_loss: 0.0134
Epoch 46/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0103 - val_loss: 0.0127
Epoch 47/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0108 - val_loss: 0.0124
Epoch 48/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0096 - val_loss: 0.0131
Epoch 49/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0099 - val_loss: 0.0115
Epoch 50/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0092 - val_loss: 0.0112
Epoch 51/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0091 - val_loss: 0.0108
Epoch 52/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0088 - val_loss: 0.0106
Epoch 53/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0086 - val_loss: 0.0107
Epoch 54/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0083 - val_loss: 0.0099
Epoch 55/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0078 - val_loss: 0.0099
Epoch 56/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0080 - val_loss: 0.0101
Epoch 57/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0080 - val_loss: 0.0095
Epoch 58/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0071 - val_loss: 0.0088
Epoch 59/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0070 - val_loss: 0.0088
Epoch 60/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0075 - val_loss: 0.0085
Epoch 61/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0066 - val_loss: 0.0091
Epoch 62/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0069 - val_loss: 0.0089
Epoch 63/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0065 - val_loss: 0.0081
Epoch 64/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0067 - val_loss: 0.0085
Epoch 65/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0064 - val_loss: 0.0082
Epoch 66/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0060 - val_loss: 0.0077
Epoch 67/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0060 - val_loss: 0.0073
Epoch 68/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0060 - val_loss: 0.0073
Epoch 69/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0055 - val_loss: 0.0068
Epoch 70/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0054 - val_loss: 0.0071
Epoch 71/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0051 - val_loss: 0.0066
Epoch 72/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0051 - val_loss: 0.0068
Epoch 73/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0050 - val_loss: 0.0063
Epoch 74/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0050 - val_loss: 0.0068
Epoch 75/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0049 - val_loss: 0.0065
```

```
Epoch 76/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0046 - val_loss: 0.0062
Epoch 77/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0044 - val_loss: 0.0065
Epoch 78/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0048 - val_loss: 0.0056
Epoch 79/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0046 - val_loss: 0.0053
Epoch 80/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0041 - val_loss: 0.0055
Epoch 81/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0038 - val_loss: 0.0059
Epoch 82/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0041 - val_loss: 0.0052
Epoch 83/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0040 - val_loss: 0.0050
Epoch 84/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0037 - val_loss: 0.0054
Epoch 85/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0035 - val_loss: 0.0047
Epoch 86/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0033 - val_loss: 0.0048
Epoch 87/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0035 - val_loss: 0.0047
Epoch 88/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0035 - val_loss: 0.0045
Epoch 89/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0032 - val_loss: 0.0047
Epoch 90/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0032 - val_loss: 0.0045
Epoch 91/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0032 - val_loss: 0.0046
Epoch 92/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0032 - val_loss: 0.0040
Epoch 93/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0029 - val_loss: 0.0042
Epoch 94/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0028 - val_loss: 0.0040
Epoch 95/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0030 - val_loss: 0.0037
Epoch 96/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0026 - val_loss: 0.0037
Epoch 97/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0025 - val_loss: 0.0038
Epoch 98/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0025 - val_loss: 0.0036
Epoch 99/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0024 - val_loss: 0.0036
Epoch 100/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0024 - val_loss: 0.0036
Epoch 101/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0024 - val_loss: 0.0034
Epoch 102/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0022 - val_loss: 0.0034
Epoch 103/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0023 - val_loss: 0.0032
Epoch 104/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0021 - val_loss: 0.0031
Epoch 105/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0019 - val_loss: 0.0029
Epoch 106/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0020 - val_loss: 0.0029
Epoch 107/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0020 - val_loss: 0.0033
Epoch 108/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0021 - val_loss: 0.0032
Epoch 109/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0021 - val_loss: 0.0032
Epoch 110/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0019 - val_loss: 0.0028
Epoch 111/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0017 - val_loss: 0.0028
```

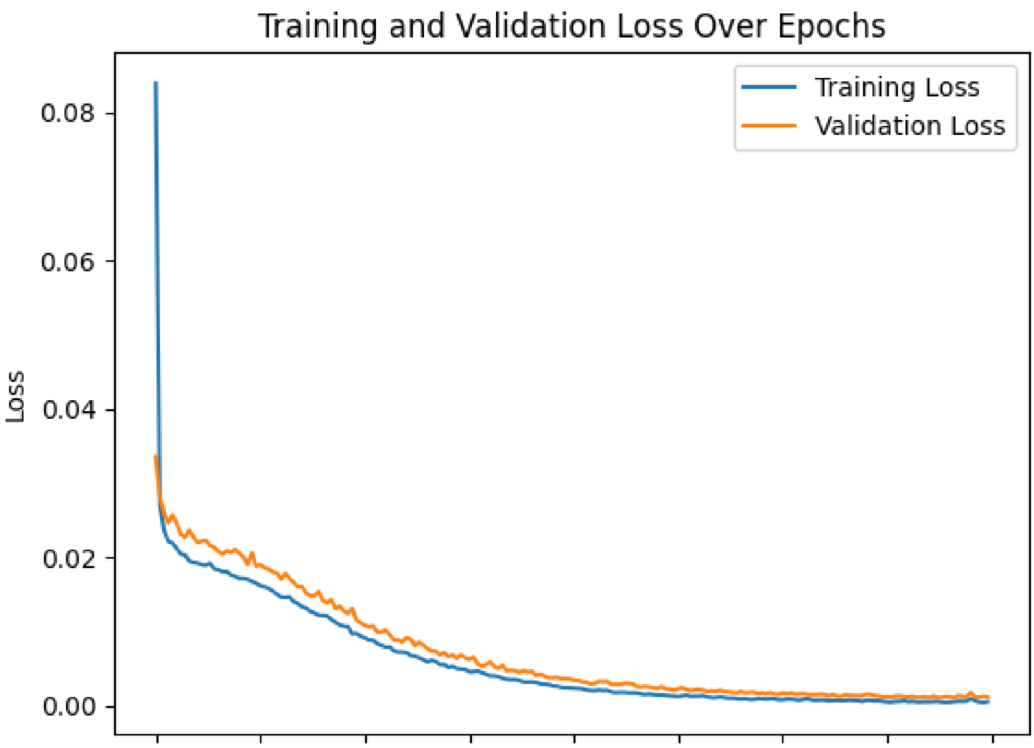
```
Epoch 112/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0017 - val_loss: 0.0028
Epoch 113/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0017 - val_loss: 0.0030
Epoch 114/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0017 - val_loss: 0.0030
Epoch 115/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0018 - val_loss: 0.0029
Epoch 116/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0017 - val_loss: 0.0026
Epoch 117/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0016 - val_loss: 0.0025
Epoch 118/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0014 - val_loss: 0.0026
Epoch 119/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0016 - val_loss: 0.0025
Epoch 120/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0014 - val_loss: 0.0024
Epoch 121/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0014 - val_loss: 0.0023
Epoch 122/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0013 - val_loss: 0.0025
Epoch 123/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0014 - val_loss: 0.0022
Epoch 124/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0013 - val_loss: 0.0022
Epoch 125/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0012 - val_loss: 0.0020
Epoch 126/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0013 - val_loss: 0.0022
Epoch 127/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0013 - val_loss: 0.0024
Epoch 128/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0014 - val_loss: 0.0021
Epoch 129/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0011 - val_loss: 0.0020
Epoch 130/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.0012 - val_loss: 0.0021
Epoch 131/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0013 - val_loss: 0.0022
Epoch 132/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0013 - val_loss: 0.0021
Epoch 133/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0012 - val_loss: 0.0019
Epoch 134/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.9283e-04 - val_loss: 0.0020
Epoch 135/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0010 - val_loss: 0.0019
Epoch 136/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0011 - val_loss: 0.0020
Epoch 137/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 0.0010 - val_loss: 0.0019
Epoch 138/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.5373e-04 - val_loss: 0.0018
Epoch 139/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.6821e-04 - val_loss: 0.0018
Epoch 140/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.2728e-04 - val_loss: 0.0017
Epoch 141/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 8.7010e-04 - val_loss: 0.0019
Epoch 142/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.4571e-04 - val_loss: 0.0017
Epoch 143/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.0325e-04 - val_loss: 0.0018
Epoch 144/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.1096e-04 - val_loss: 0.0018
Epoch 145/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.4492e-04 - val_loss: 0.0016
Epoch 146/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.8446e-04 - val_loss: 0.0017
Epoch 147/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.6038e-04 - val_loss: 0.0015
```

```
Epoch 148/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.8477e-04 - val_loss: 0.0017
Epoch 149/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 9.6235e-04 - val_loss: 0.0015
Epoch 150/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 7.4591e-04 - val_loss: 0.0015
Epoch 151/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.2462e-04 - val_loss: 0.0017
Epoch 152/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.0756e-04 - val_loss: 0.0015
Epoch 153/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.1992e-04 - val_loss: 0.0017
Epoch 154/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.9153e-04 - val_loss: 0.0015
Epoch 155/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.0745e-04 - val_loss: 0.0016
Epoch 156/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 8.1839e-04 - val_loss: 0.0016
Epoch 157/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 8.5888e-04 - val_loss: 0.0014
Epoch 158/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.8904e-04 - val_loss: 0.0015
Epoch 159/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.7370e-04 - val_loss: 0.0014
Epoch 160/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.8719e-04 - val_loss: 0.0015
Epoch 161/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.9943e-04 - val_loss: 0.0015
Epoch 162/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.3656e-04 - val_loss: 0.0013
Epoch 163/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.5988e-04 - val_loss: 0.0015
Epoch 164/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.0092e-04 - val_loss: 0.0013
Epoch 165/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.3000e-04 - val_loss: 0.0012
Epoch 166/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.8499e-04 - val_loss: 0.0014
Epoch 167/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.5915e-04 - val_loss: 0.0013
Epoch 168/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.6432e-04 - val_loss: 0.0013
Epoch 169/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.5779e-04 - val_loss: 0.0013
Epoch 170/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.7199e-04 - val_loss: 0.0013
Epoch 171/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.4041e-04 - val_loss: 0.0015
Epoch 172/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.0442e-04 - val_loss: 0.0015
Epoch 173/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.2694e-04 - val_loss: 0.0013
Epoch 174/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 6.4703e-04 - val_loss: 0.0013
Epoch 175/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.2022e-04 - val_loss: 0.0011
Epoch 176/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6660e-04 - val_loss: 0.0012
Epoch 177/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7339e-04 - val_loss: 0.0011
Epoch 178/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.3330e-04 - val_loss: 0.0013
Epoch 179/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.5705e-04 - val_loss: 0.0013
Epoch 180/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.8811e-04 - val_loss: 0.0012
Epoch 181/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.3013e-04 - val_loss: 0.0011
Epoch 182/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 5.6960e-04 - val_loss: 0.0012
Epoch 183/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.2001e-04 - val_loss: 0.0011
```

```
Epoch 184/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.6059e-04 - val_loss: 0.0011
Epoch 185/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.8554e-04 - val_loss: 0.0012
Epoch 186/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.8546e-04 - val_loss: 0.0010
Epoch 187/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.1965e-04 - val_loss: 0.0013
Epoch 188/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.9058e-04 - val_loss: 0.0010
Epoch 189/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.9747e-04 - val_loss: 0.0011
Epoch 190/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.8539e-04 - val_loss: 0.0012
Epoch 191/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7775e-04 - val_loss: 0.0011
Epoch 192/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0961e-04 - val_loss: 0.0010
Epoch 193/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.9932e-04 - val_loss: 0.0013
Epoch 194/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.7334e-04 - val_loss: 0.0012
Epoch 195/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.7393e-04 - val_loss: 0.0012
Epoch 196/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.3511e-04 - val_loss: 0.0017
Epoch 197/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 7.9082e-04 - val_loss: 0.0012
Epoch 198/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 5.0701e-04 - val_loss: 0.0011
Epoch 199/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.3866e-04 - val_loss: 0.0012
Epoch 200/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 4.7701e-04 - val_loss: 0.0012
```

In [18]:

```python
plt.plot(historyRnn.history['loss'], label='Training Loss')
# Plot validation loss
plt.plot(historyRnn.history['val_loss'], label='Validation Loss')

plt.title('Training and Validation Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



In [19]:

```
y_pred_scaled = modelRnn.predict(X_test)
y_pred_actual = scaler.inverse_transform(y_pred_scaled)
y_test_actual = scaler.inverse_transform(y_test)
```

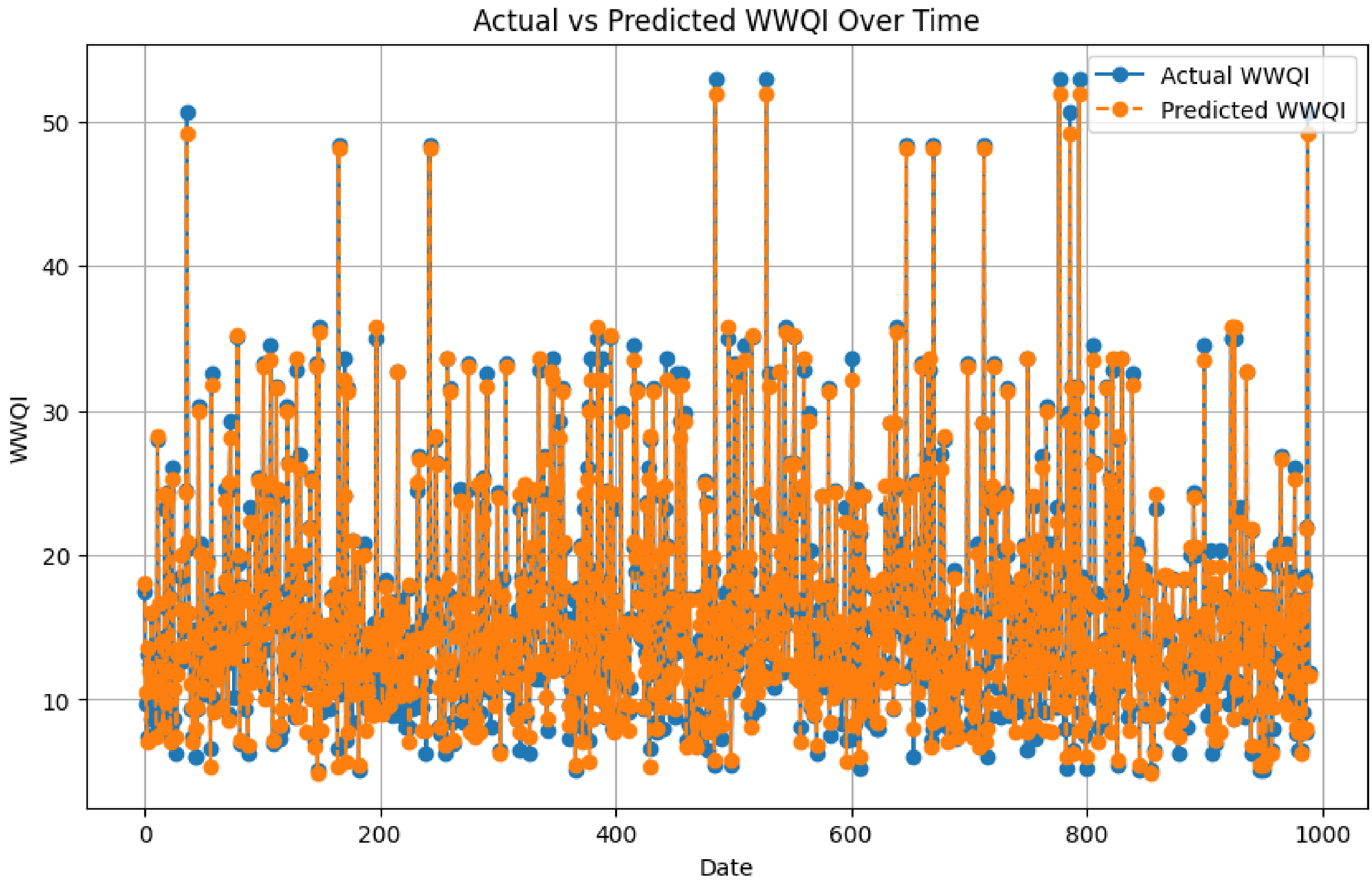
```
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step
```

In [20]:

```
actual_values = y_test_actual[:, 7]
predicted_values = y_pred_actual[:, 7]
```

In [23]:

```
plt.figure(figsize=(10, 6))
plt.plot(actual_values, label='Actual WWQI', marker='o')
plt.plot(predicted_values, label='Predicted WWQI', linestyle='dashed', marker='o')

plt.title('Actual vs Predicted WWQI Over Time')
plt.xlabel('Date')
plt.ylabel('WWQI')
plt.legend()
plt.grid(True)
plt.show()
```



In [24]:

```
# Calculate MAE and MSE
rnn_mae = mean_absolute_error(actual_values, predicted_values)
mse = mean_squared_error(actual_values, predicted_values)
rnn_rmse = np.sqrt(mse)
r_squared_rnn = r2_score(actual_values, predicted_values)

print(f'Mean Absolute Error (MAE): {rnn_mae}')
print(f'Root Mean Squared Error (RMSE): {rnn_rmse}')
```

```
print("R-squared value:", r_squared_rnn)
```

```
Mean Absolute Error (MAE): 0.7502541925305234
Root Mean Squared Error (RMSE): 0.9846687854627356
R-squared value: 0.9844604460744509
```

In [25]:

```
future_steps_rnn = 120  # 10 years * 12 months

# Initial sequence to start prediction
initial_sequence_rnn = scaled_data[-seq_length:]

# Predict the future values
future_predictions_scaled_rnn = []

for _ in range(future_steps_rnn):
    next_pred_scaled_rnn = modelRnn.predict(initial_sequence_rnn.reshape(1, seq_length,
scaled_data.shape[1]))
    future_predictions_scaled_rnn.append(next_pred_scaled_rnn)
    initial_sequence_rnn = np.concatenate((initial_sequence_rnn[1:], next_pred_scaled_rn
n), axis=0)

# Convert the predictions to array
future_predictions_scaled_rnn = np.array(future_predictions_scaled_rnn).squeeze()

# Inverse transform the predictions to get them in the original scale
future_predictions_rnn = scaler.inverse_transform(future_predictions_scaled_rnn)
```

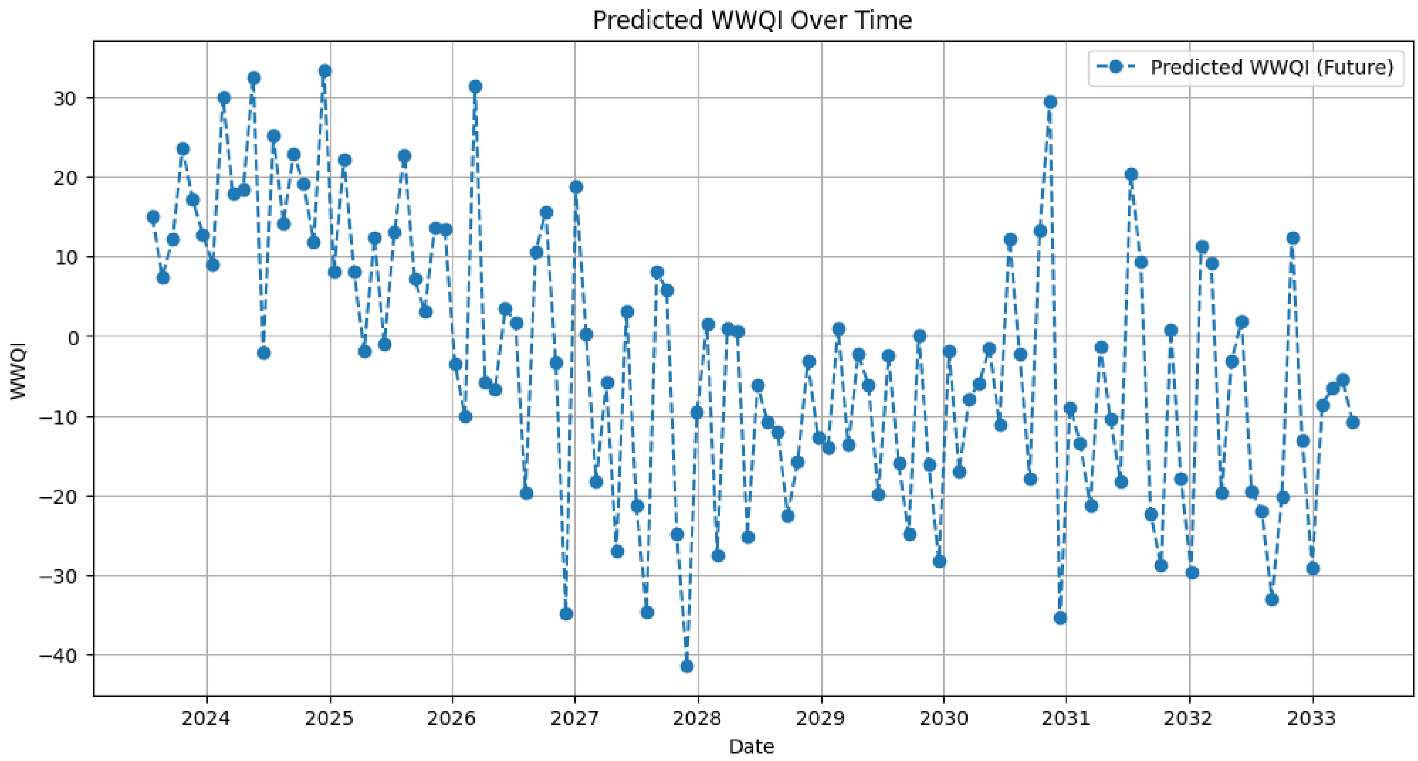
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

In [26]:

```
import datetime
last_date_rnn = dfs.index[-1]

# Generate future dates for the extended time series
future_dates_rnn = [last_date_rnn + datetime.timedelta(days=i * 30) for i in range(1, fu
ture_steps_rnn + 1)]

# Plot the predicted WWQI values for the future
plt.figure(figsize=(12, 6))
plt.plot(future_dates_rnn, future_predictions_rnn[:, 7], label='Predicted WWQI (Future)',
linestyle='dashed', marker='o')

plt.title('Predicted WWQI Over Time')
plt.xlabel('Date')
plt.ylabel('WWQI')
plt.legend()
plt.grid(True)
plt.show()
```



In [27]:

```
dfr = df.copy()
```

In [28]:

```
dfr = dfr.drop('WWQIs', axis=1)
```

In [29]:

```
dfr.head()
```

Out[29]:

	Date	pH	Temp	TDS	TSS	Cl	BOD	COD	WWQIi
0	2021-07-02	7.6	30.0	316.0	18.0	76.0	1.5	33.6	13.65
1	2021-07-05	7.2	29.0	240.0	36.0	43.0	1.5	8.4	9.10

	Date	pH	Temp	TDS	TSS	Cl	BOD	COD	WWQIi
2	2021-07-07	7.3	30.0	276.0	21.0	51.0	0.8	16.8	9.28
3	2021-07-09	7.1	28.0	194.0	15.0	32.0	1.4	8.4	5.39
4	2021-07-12	7.5	29.0	198.0	16.0	36.0	1.1	8.4	10.00

In [30]:

```
dfr['Date'] = pd.to_datetime(dfr['Date'])
dfr.set_index('Date', inplace=True)
```

In [31]:

```
null_percentage = (dfr.isnull().sum() / len(dfr)) * 100
null_percentage
```

Out[31]:

```
pH       0.347222
Temp     0.347222
TDS      0.347222
TSS      0.347222
Cl       0.347222
BOD      0.347222
COD      0.347222
WWQIi    0.347222
dtype: float64
```

In [32]:

```
for column in dfr.columns.difference(['Date']):
    mean_value = dfr[column].mean()
    dfr[column].fillna(mean_value, inplace=True)
```

```
/tmp/ipykernel_33/4147450775.py:3: FutureWarning: A value is trying to be set on a copy o
f a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the i
ntermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col:
value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operati
on inplace on the original object.


  dfr[column].fillna(mean_value, inplace=True)
```

In [33]:

```
null_percentage = (dfr.isnull().sum() / len(dfr)) * 100
null_percentage
```

Out[33]:

```
pH       0.0
Temp     0.0
TDS      0.0
TSS      0.0
Cl       0.0
BOD      0.0
COD      0.0
WWQIi    0.0
dtype: float64
```

In [34]:

```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(dfr)
```

In [41]:

```
def create_sequences(data, seq_length):
```

```
    sequences = []
    targets = []

    for i in range(len(data) - seq_length):
        seq = data[i:i + seq_length]
        label = data[i + seq_length]
        sequences.append(seq)
        targets.append(label)

    return np.array(sequences), np.array(targets)

seq_length = 10  # You can adjust this based on your needs
X, y = create_sequences(scaled_data, seq_length)
```

In [42]:

```
X.shape
```

Out[42]:

```
(278, 10, 8)
```

In [43]:

```
y.shape
```

Out[43]:

```
(278, 8)
```

In [44]:

```
def custom_train_test_split(X, y, test_size=0.2, random_state=None):
    classes = np.unique(y)
    train_indices, test_indices = [], []
    for c in classes:
        indices = np.where(y == c)[0]
        np.random.shuffle(indices)
        split_idx = int(len(indices) * (1 - test_size))
        train_indices.extend(indices[:split_idx])
        test_indices.extend(indices[split_idx:])
    np.random.shuffle(train_indices)
    np.random.shuffle(test_indices)
    X_train, X_test = X[train_indices], X[test_indices]
    y_train, y_test = y[train_indices], y[test_indices]
    return X_train, X_test, y_train, y_test

X_train_lstm, X_test_lstm, y_train_lstm, y_test_lstm = custom_train_test_split(X, y, tes
t_size=0.2, random_state=42)
```

In [45]:

```
X_train.shape
```

Out[45]:

```
(1234, 10, 8)
```

In [ ]:

```

```

LSTM Modelling

In [ ]:

```

```

In [47]:

```
model1 = Sequential()
```

```
model1 = Sequential()
model1.add(LSTM(units=16, return_sequences=True, input_shape=(X.shape[1], X.shape[2])))
model1.add(LSTM(units=64))

model1.add(Dense(units=scaled_data.shape[1])) # Assuming the output size is the same as i
nput size

model1.compile(optimizer='adam', loss='mean_squared_error')

history1 = model1.fit(X_train_lstm, y_train_lstm, epochs=200, batch_size=24, validation_
data=(X_test_lstm, y_test_lstm), verbose=1)
```

```
Epoch 1/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 4s 18ms/step - loss: 0.0518 - val_loss: 0.0277
Epoch 2/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0236 - val_loss: 0.0263
Epoch 3/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0215 - val_loss: 0.0252
Epoch 4/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0208 - val_loss: 0.0257
Epoch 5/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0211 - val_loss: 0.0247
Epoch 6/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0205 - val_loss: 0.0244
Epoch 7/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0209 - val_loss: 0.0241
Epoch 8/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0191 - val_loss: 0.0238
Epoch 9/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0199 - val_loss: 0.0240
Epoch 10/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0193 - val_loss: 0.0236
Epoch 11/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0198 - val_loss: 0.0237
Epoch 12/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0203 - val_loss: 0.0237
Epoch 13/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0187 - val_loss: 0.0236
Epoch 14/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0199 - val_loss: 0.0242
Epoch 15/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0190 - val_loss: 0.0238
Epoch 16/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0197 - val_loss: 0.0234
Epoch 17/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0189 - val_loss: 0.0230
Epoch 18/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0187 - val_loss: 0.0226
Epoch 19/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0174 - val_loss: 0.0225
Epoch 20/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0190 - val_loss: 0.0228
Epoch 21/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0199 - val_loss: 0.0230
Epoch 22/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0182 - val_loss: 0.0232
Epoch 23/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0179 - val_loss: 0.0233
Epoch 24/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0189 - val_loss: 0.0229
Epoch 25/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0178 - val_loss: 0.0229
Epoch 26/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0182 - val_loss: 0.0220
Epoch 27/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0182 - val_loss: 0.0225
Epoch 28/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0179 - val_loss: 0.0221
Epoch 29/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0176 - val_loss: 0.0221
Epoch 30/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0168 - val_loss: 0.0221
Epoch 31/200
```

```
Epoch 31/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0178 - val_loss: 0.0219
Epoch 32/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0176 - val_loss: 0.0221
Epoch 33/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0179 - val_loss: 0.0218
Epoch 34/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0167 - val_loss: 0.0228
Epoch 35/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0161 - val_loss: 0.0213
Epoch 36/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0182 - val_loss: 0.0211
Epoch 37/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0175 - val_loss: 0.0212
Epoch 38/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0163 - val_loss: 0.0216
Epoch 39/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0169 - val_loss: 0.0208
Epoch 40/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0165 - val_loss: 0.0208
Epoch 41/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0161 - val_loss: 0.0206
Epoch 42/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0160 - val_loss: 0.0205
Epoch 43/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0169 - val_loss: 0.0208
Epoch 44/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0157 - val_loss: 0.0204
Epoch 45/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0161 - val_loss: 0.0200
Epoch 46/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0150 - val_loss: 0.0203
Epoch 47/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0151 - val_loss: 0.0198
Epoch 48/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0155 - val_loss: 0.0204
Epoch 49/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0155 - val_loss: 0.0194
Epoch 50/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 0.0147 - val_loss: 0.0193
Epoch 51/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0152 - val_loss: 0.0189
Epoch 52/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0139 - val_loss: 0.0192
Epoch 53/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0148 - val_loss: 0.0193
Epoch 54/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0153 - val_loss: 0.0184
Epoch 55/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0140 - val_loss: 0.0185
Epoch 56/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0141 - val_loss: 0.0181
Epoch 57/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0143 - val_loss: 0.0177
Epoch 58/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0142 - val_loss: 0.0180
Epoch 59/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0136 - val_loss: 0.0180
Epoch 60/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0139 - val_loss: 0.0173
Epoch 61/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0123 - val_loss: 0.0172
Epoch 62/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0132 - val_loss: 0.0169
Epoch 63/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0131 - val_loss: 0.0170
Epoch 64/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0130 - val_loss: 0.0163
Epoch 65/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0130 - val_loss: 0.0165
Epoch 66/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0132 - val_loss: 0.0162
Epoch 67/200
```

```
Epoch 67/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0129 - val_loss: 0.0161
Epoch 68/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0121 - val_loss: 0.0160
Epoch 69/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0124 - val_loss: 0.0151
Epoch 70/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0113 - val_loss: 0.0150
Epoch 71/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0120 - val_loss: 0.0150
Epoch 72/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0116 - val_loss: 0.0147
Epoch 73/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0111 - val_loss: 0.0146
Epoch 74/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0111 - val_loss: 0.0141
Epoch 75/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0112 - val_loss: 0.0145
Epoch 76/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0112 - val_loss: 0.0142
Epoch 77/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0110 - val_loss: 0.0140
Epoch 78/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0109 - val_loss: 0.0138
Epoch 79/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0108 - val_loss: 0.0132
Epoch 80/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0099 - val_loss: 0.0132
Epoch 81/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0105 - val_loss: 0.0130
Epoch 82/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0099 - val_loss: 0.0122
Epoch 83/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0095 - val_loss: 0.0123
Epoch 84/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0093 - val_loss: 0.0125
Epoch 85/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0092 - val_loss: 0.0120
Epoch 86/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0095 - val_loss: 0.0114
Epoch 87/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0090 - val_loss: 0.0111
Epoch 88/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0089 - val_loss: 0.0108
Epoch 89/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0087 - val_loss: 0.0109
Epoch 90/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0088 - val_loss: 0.0107
Epoch 91/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0095 - val_loss: 0.0102
Epoch 92/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0080 - val_loss: 0.0106
Epoch 93/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0083 - val_loss: 0.0101
Epoch 94/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0075 - val_loss: 0.0096
Epoch 95/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0080 - val_loss: 0.0108
Epoch 96/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0081 - val_loss: 0.0094
Epoch 97/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0075 - val_loss: 0.0092
Epoch 98/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0073 - val_loss: 0.0088
Epoch 99/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0073 - val_loss: 0.0086
Epoch 100/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0071 - val_loss: 0.0085
Epoch 101/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0067 - val_loss: 0.0083
Epoch 102/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0065 - val_loss: 0.0082
Epoch 103/200
```

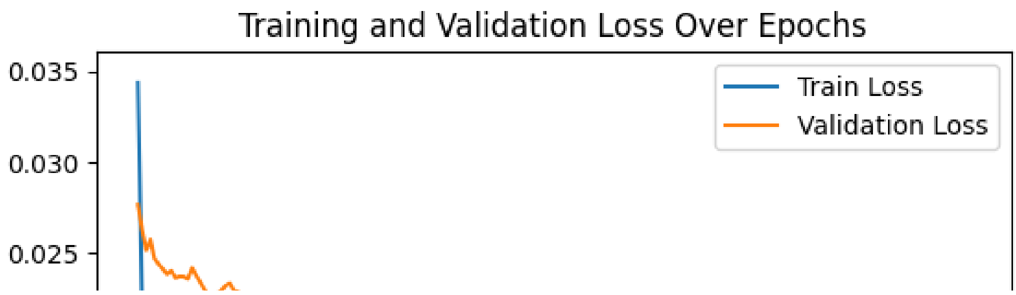
```
Epoch 103/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step - loss: 0.0065 - val_loss: 0.0077
Epoch 104/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0060 - val_loss: 0.0075
Epoch 105/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0068 - val_loss: 0.0097
Epoch 106/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0070 - val_loss: 0.0073
Epoch 107/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0059 - val_loss: 0.0070
Epoch 108/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0055 - val_loss: 0.0066
Epoch 109/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0052 - val_loss: 0.0062
Epoch 110/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0058 - val_loss: 0.0068
Epoch 111/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0054 - val_loss: 0.0057
Epoch 112/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0050 - val_loss: 0.0056
Epoch 113/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0047 - val_loss: 0.0054
Epoch 114/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0045 - val_loss: 0.0053
Epoch 115/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0048 - val_loss: 0.0054
Epoch 116/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0049 - val_loss: 0.0051
Epoch 117/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0047 - val_loss: 0.0051
Epoch 118/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0044 - val_loss: 0.0046
Epoch 119/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0041 - val_loss: 0.0046
Epoch 120/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0040 - val_loss: 0.0043
Epoch 121/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0039 - val_loss: 0.0043
Epoch 122/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0037 - val_loss: 0.0040
Epoch 123/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0036 - val_loss: 0.0039
Epoch 124/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0035 - val_loss: 0.0042
Epoch 125/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0033 - val_loss: 0.0037
Epoch 126/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0034 - val_loss: 0.0038
Epoch 127/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0034 - val_loss: 0.0037
Epoch 128/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0034 - val_loss: 0.0035
Epoch 129/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0032 - val_loss: 0.0038
Epoch 130/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0033 - val_loss: 0.0037
Epoch 131/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0031 - val_loss: 0.0032
Epoch 132/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0029 - val_loss: 0.0035
Epoch 133/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0030 - val_loss: 0.0032
Epoch 134/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0027 - val_loss: 0.0031
Epoch 135/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0029 - val_loss: 0.0031
Epoch 136/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 137/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0025 - val_loss: 0.0028
Epoch 138/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0025 - val_loss: 0.0028
Epoch 139/200
```

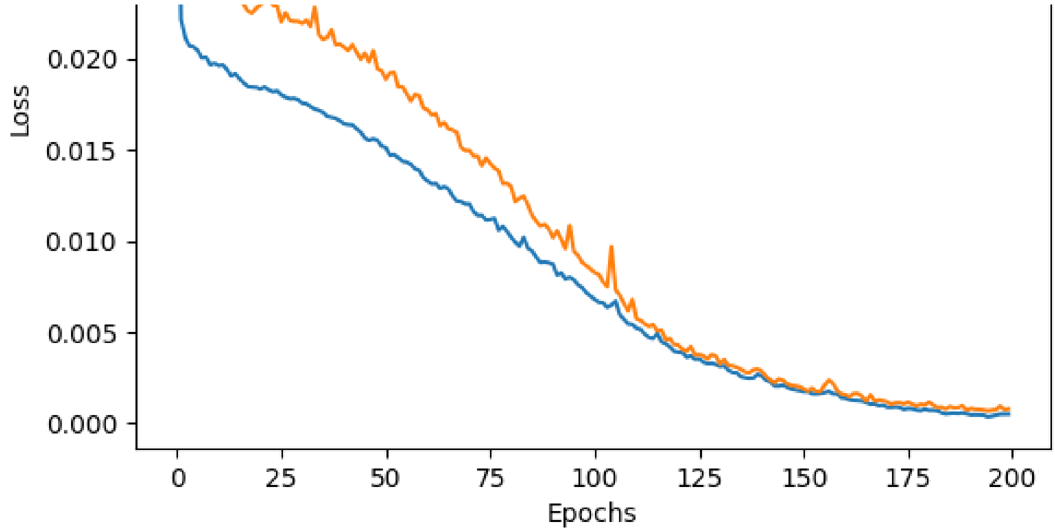
```
Epoch 139/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0025 - val_loss: 0.0029
Epoch 140/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0025 - val_loss: 0.0030
Epoch 141/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0027 - val_loss: 0.0028
Epoch 142/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0025 - val_loss: 0.0026
Epoch 143/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0023 - val_loss: 0.0024
Epoch 144/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0020 - val_loss: 0.0022
Epoch 145/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0020 - val_loss: 0.0024
Epoch 146/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0021 - val_loss: 0.0023
Epoch 147/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0020 - val_loss: 0.0022
Epoch 148/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0019 - val_loss: 0.0021
Epoch 149/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0019 - val_loss: 0.0020
Epoch 150/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0016 - val_loss: 0.0020
Epoch 151/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0018 - val_loss: 0.0019
Epoch 152/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0018 - val_loss: 0.0018
Epoch 153/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0016 - val_loss: 0.0019
Epoch 154/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.0015 - val_loss: 0.0018
Epoch 155/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0016 - val_loss: 0.0017
Epoch 156/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 0.0017 - val_loss: 0.0021
Epoch 157/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0017 - val_loss: 0.0024
Epoch 158/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0017 - val_loss: 0.0022
Epoch 159/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0017 - val_loss: 0.0018
Epoch 160/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0015 - val_loss: 0.0016
Epoch 161/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0014 - val_loss: 0.0015
Epoch 162/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0012 - val_loss: 0.0015
Epoch 163/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0012 - val_loss: 0.0016
Epoch 164/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0012 - val_loss: 0.0016
Epoch 165/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0012 - val_loss: 0.0015
Epoch 166/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0012 - val_loss: 0.0012
Epoch 167/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0010 - val_loss: 0.0015
Epoch 168/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0011 - val_loss: 0.0012
Epoch 169/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 9.2668e-04 - val_loss: 0.0013
Epoch 170/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 0.0011 - val_loss: 0.0012
Epoch 171/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.8007e-04 - val_loss: 0.0012
Epoch 172/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - loss: 8.5549e-04 - val_loss: 0.0010
Epoch 173/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 9.2492e-04 - val_loss: 0.0011
Epoch 174/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 9.0769e-04 - val_loss: 0.0011
Epoch 175/200
```

```
Epoch 175/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 8.1504e-04 - val_loss: 0.0011
Epoch 176/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.4995e-04 - val_loss: 0.0012
Epoch 177/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.6929e-04 - val_loss: 0.0010
Epoch 178/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.2338e-04 - val_loss: 9.4138e-04
Epoch 179/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 6.2056e-04 - val_loss: 0.0010
Epoch 180/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.2586e-04 - val_loss: 9.9988e-04
Epoch 181/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 6.2678e-04 - val_loss: 0.0012
Epoch 182/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 6.8517e-04 - val_loss: 9.9395e-04
Epoch 183/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 7.1418e-04 - val_loss: 8.5484e-04
Epoch 184/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 5.7941e-04 - val_loss: 8.8057e-04
Epoch 185/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 5.4712e-04 - val_loss: 7.7263e-04
Epoch 186/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 5.1311e-04 - val_loss: 9.1209e-04
Epoch 187/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 5.3465e-04 - val_loss: 8.4344e-04
Epoch 188/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 5.2972e-04 - val_loss: 8.4593e-04
Epoch 189/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 5.5813e-04 - val_loss: 9.5904e-04
Epoch 190/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 5.7014e-04 - val_loss: 7.1329e-04
Epoch 191/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 4.2473e-04 - val_loss: 8.0449e-04
Epoch 192/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 4.6525e-04 - val_loss: 7.5490e-04
Epoch 193/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 4.5430e-04 - val_loss: 7.4787e-04
Epoch 194/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 4.9162e-04 - val_loss: 7.2000e-04
Epoch 195/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 3.4341e-04 - val_loss: 6.7324e-04
Epoch 196/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 3.8362e-04 - val_loss: 7.0952e-04
Epoch 197/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 4.3810e-04 - val_loss: 7.3240e-04
Epoch 198/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 4.7561e-04 - val_loss: 9.4372e-04
Epoch 199/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 5.4546e-04 - val_loss: 7.3578e-04
Epoch 200/200
52/52 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - loss: 4.8420e-04 - val_loss: 7.7339e-04
```

In [48]:

```python
plt.plot(history1.history['loss'], label='Train Loss')
plt.plot(history1.history['val_loss'], label='Validation Loss')
plt.legend()
plt.title('Training and Validation Loss Over Epochs')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```





In [49]:

```
y_pred_scaled = model1.predict(X_test)
y_pred_actual = scaler.inverse_transform(y_pred_scaled)
y_test_actual = scaler.inverse_transform(y_test)
```

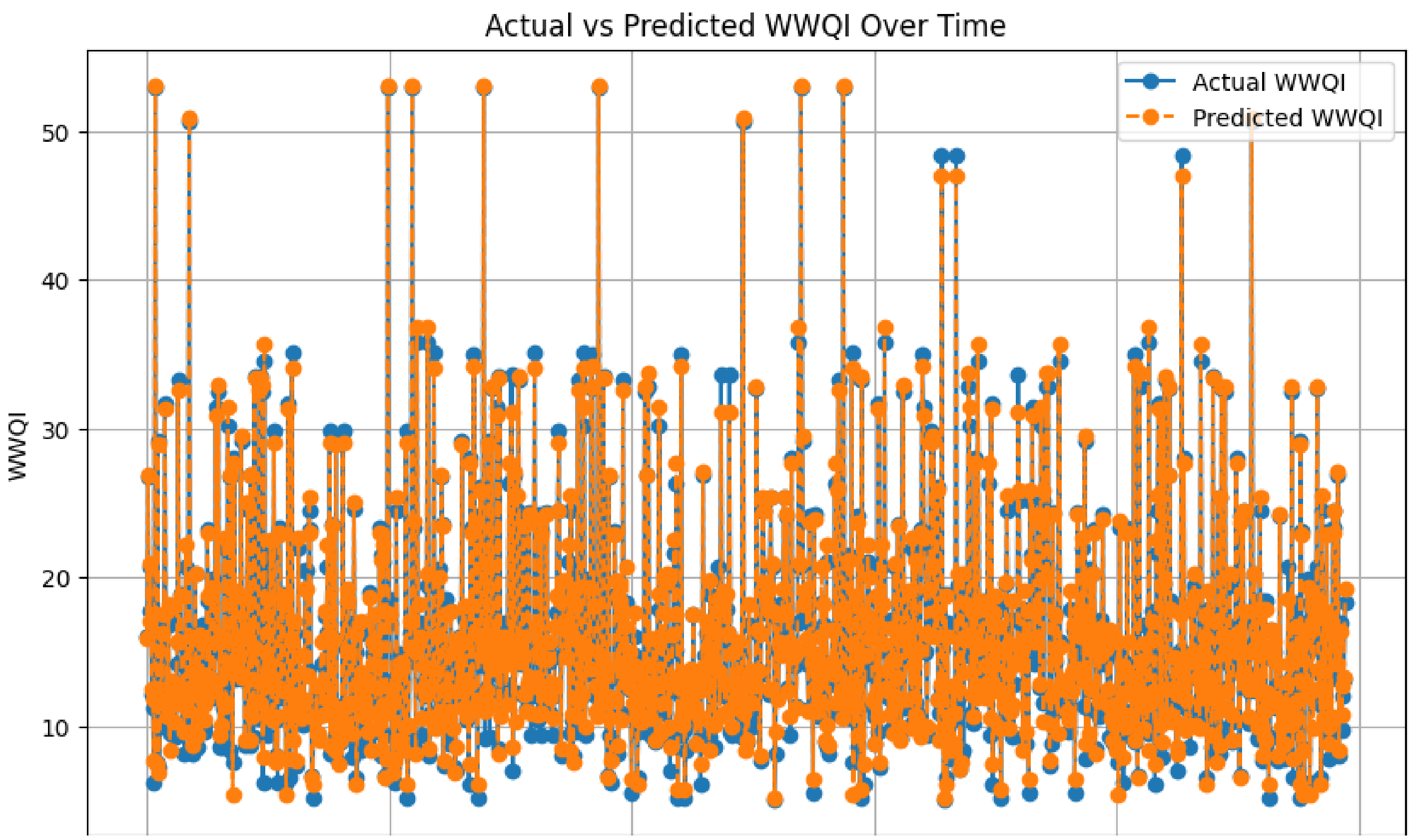
```
31/31 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step
```

In [50]:

```
actual_values_lstm = y_test_actual[:, 7]
predicted_values_lstm = y_pred_actual[:, 7]
```

In [51]:

```
plt.figure(figsize=(10, 6))
plt.plot(actual_values_lstm, label='Actual WWQI', marker='o')
plt.plot(predicted_values_lstm, label='Predicted WWQI', linestyle='dashed', marker='o')

plt.title('Actual vs Predicted WWQI Over Time')
plt.xlabel('Date')
plt.ylabel('WWQI')
plt.legend()
plt.grid(True)
plt.show()
```



In [52]:

```
# Calculate MAE and MSE
lstm_mae = mean_absolute_error(actual_values_lstm, predicted_values_lstm)
mse = mean_squared_error(actual_values_lstm, predicted_values_lstm)
lstm_rmse = np.sqrt(mse)
r_squared_lstm = r2_score(actual_values_lstm, predicted_values_lstm)

print(f'Mean Absolute Error (MAE): {lstm_mae}')
print(f'Root Mean Squared Error (RMSE): {lstm_rmse}')
print("R-squared value:", r_squared_lstm)
```

```
Mean Absolute Error (MAE): 0.7527528116679119
Root Mean Squared Error (RMSE): 1.0600869985501247
R-squared value: 0.9828037027096039
```

In [38]:

```

```

```
R-squared value: 0.9773361364682523
```

In [75]:

```
future_steps = 120  # 10 years * 12 months

# Initial sequence to start prediction
initial_sequence = scaled_data[-seq_length:]

# Predict the future values
future_predictions_scaled = []

for _ in range(future_steps):
    next_pred_scaled = model1.predict(initial_sequence.reshape(1, seq_length, scaled_dat
a.shape[1]))
    future_predictions_scaled.append(next_pred_scaled)
    initial_sequence = np.concatenate((initial_sequence[1:], next_pred_scaled), axis=0)

# Convert the predictions to array
future_predictions_scaled = np.array(future_predictions_scaled).squeeze()

# Inverse transform the predictions to get them in the original scale
future_predictions = scaler.inverse_transform(future_predictions_scaled)
```

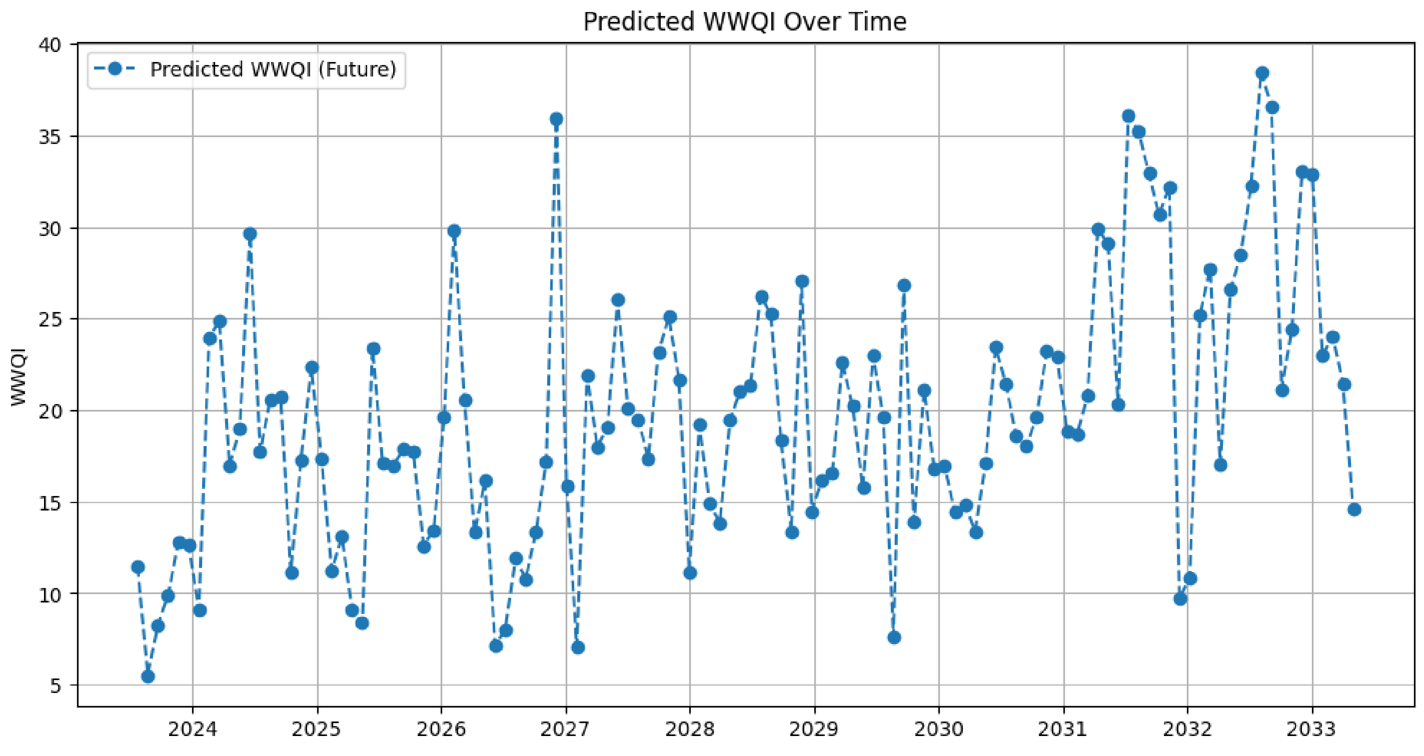
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
```

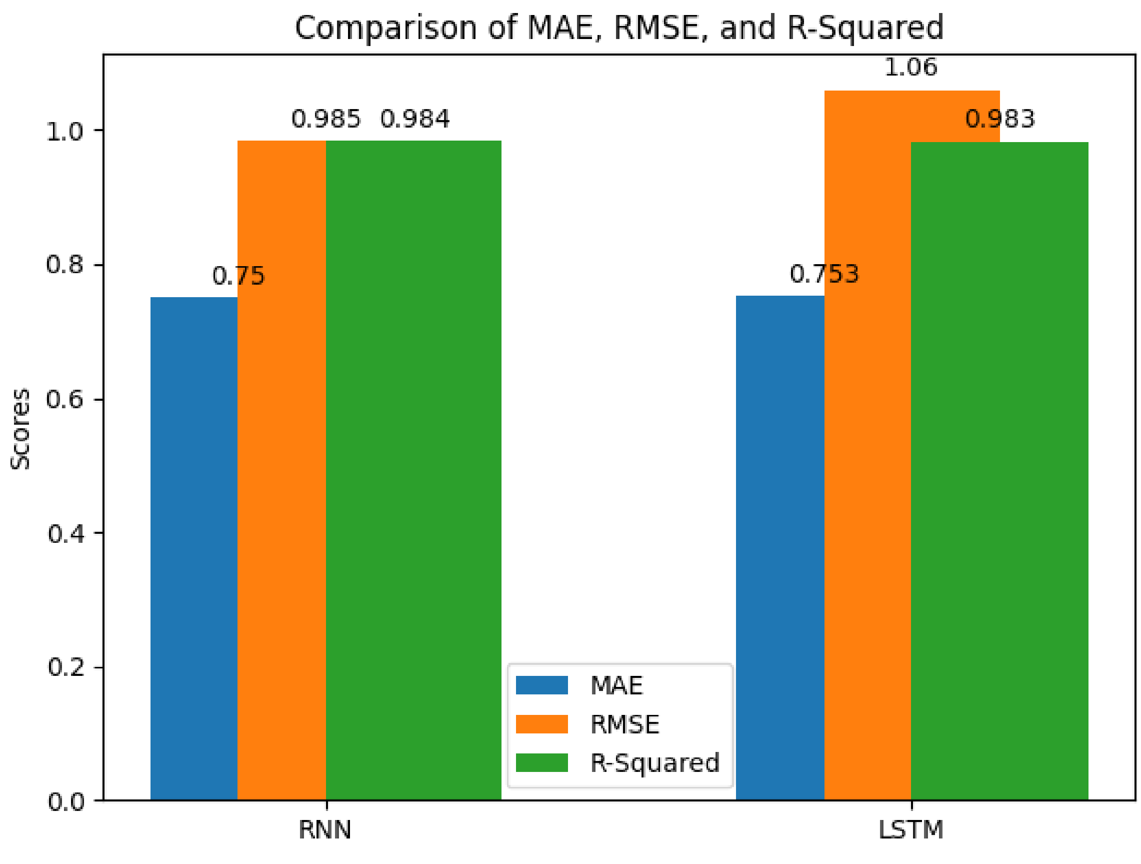
In [96]:

```python
import datetime

# Assuming 'dfs' is your original DataFrame with 'Date' as index
last_date = dfr.index[-1]

# Generate future dates for the extended time series
future_dates = [last_date + datetime.timedelta(days=i * 30) for i in range(1, future_ste
ps + 1)]

# Plot the predicted WWQI values for the future
plt.figure(figsize=(12, 6))
plt.plot(future_dates, future_predictions[:, 7], label='Predicted WWQI (Future)', linesty
le='dashed', marker='o')

plt.title('Predicted WWQI Over Time')
plt.xlabel('Date')
plt.ylabel('WWQI')
plt.legend()
plt.grid(True)
plt.show()
```



In [62]:

```
import numpy as np
import matplotlib.pyplot as plt

labels = ['RNN', 'LSTM']
mae_values = [round(rnn_mae, 3), round(lstm_mae, 3)]
rmse_values = [round(rnn_rmse, 3), round(lstm_rmse, 3)]
r2_values = [round(r_squared_rnn, 3), round(r_squared_lstm, 3)]
x = np.arange(len(labels))
width = 0.3

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, mae_values, width, label='MAE')
rects2 = ax.bar(x, rmse_values, width, label='RMSE')
rects3 = ax.bar(x + width/2, r2_values, width, label='R-Squared')

ax.set_ylabel('Scores')
ax.set_title('Comparison of MAE, RMSE, and R-Squared')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()

# Function to add labels on top of the bars
def autolabel(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate('{}'.format(height),
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),  # 3 points vertical offset
                    textcoords="offset points",
                    ha='center', va='bottom')

autolabel(rects1)
autolabel(rects2)
autolabel(rects3)

fig.tight_layout()
plt.savefig('rnn_vs_lstm.png')
plt.show()
```



WWQI Predictor

In [91]:

```
import numpy as np
import datetime

# Define the last date in your initial data
initial_year = 2023
initial_month = 4
initial_day = 6
initial_date = datetime.datetime(year=initial_year, month=initial_month, day=initial_day)
year = int(input("Enter the year: "))
month = int(input("Enter the month: "))
day = int(input("Enter the day: "))

# Define the target date
target_date = datetime.datetime(year=year, month=month, day=day)

# Find the index corresponding to the target date in the future dates list
delta_days = (target_date - last_date).days

# Calculate the index in the future predictions array
index = delta_days // 30  # Assuming each step in future_dates corresponds to 30 days

# Check if the index is within the range of available predictions
if 0 <= index < len(future_predictions):
    # Get the predicted value for the target date
    target_prediction = future_predictions[index, 7]  # Assuming you want the prediction
for the 7th column
    print("Prediction for {}: {}".format(target_date.strftime("%Y-%m-%d"), target_predic
tion))
else:
    print("Prediction is not available for the given date.")
```

```
Prediction for 2032-01-05: 10.84066390991211
```

In [ ]:

```

```