In [10]:
```
import math
import yfinance as yf
import numpy as np
import pandas as pd
from sklearn.preprocessing import  MinMaxScaler
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

stock_data=yf.download('MSFT',start='2016-01-01',end='2021-10-01')
stock_data.head()
```

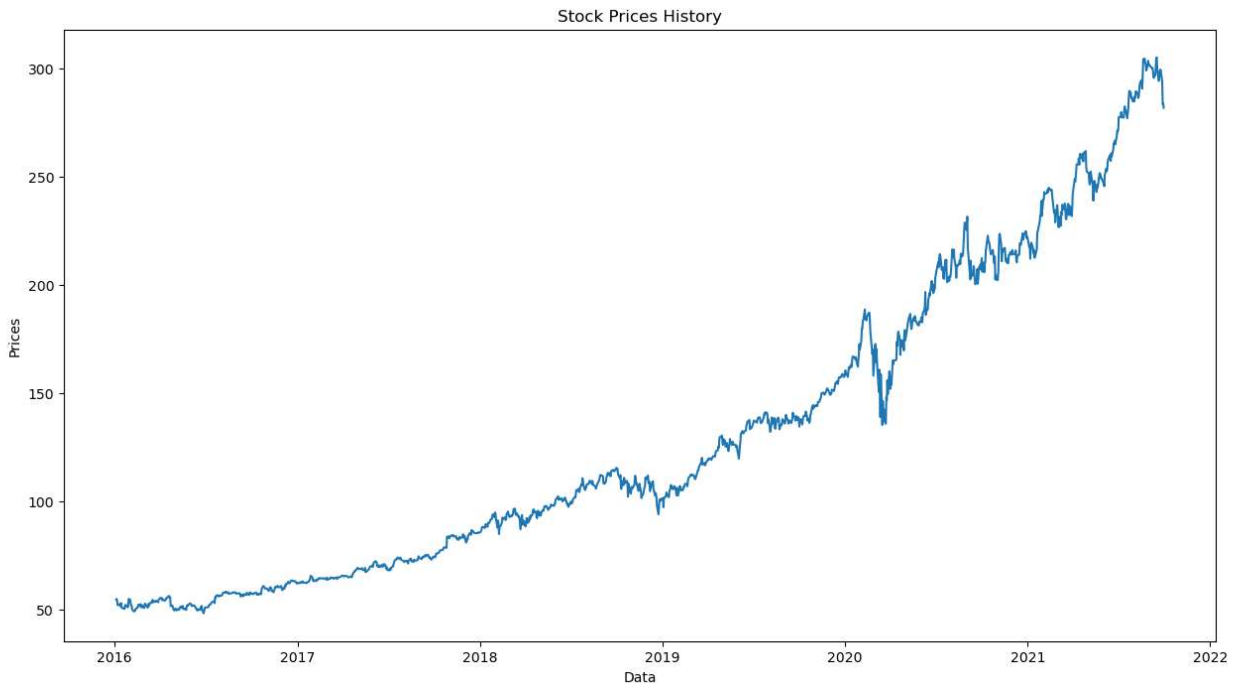
```
[*********************100%***********************]  1 of 1 completed
```

Out[10]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2016-01-04	54.320000	54.799999	53.389999	54.799999	48.901043	53778000
2016-01-05	54.930000	55.389999	54.540001	55.049999	49.124138	34079700
2016-01-06	54.320000	54.400002	53.639999	54.049999	48.231770	39518900
2016-01-07	52.700001	53.490002	52.070000	52.169998	46.554150	56564900
2016-01-08	52.369999	53.279999	52.150002	52.330002	46.696934	48754000

In [12]:
```
plt.figure(figsize=(15,8))
plt.title('Stock Prices History')
plt.plot(stock_data['Close'])
plt.xlabel('Data')
plt.ylabel('Prices')
```

Out[12]:
```
Text(0, 0.5, 'Prices')
```



In [16]:
```
close_prices = stock_data['Close']
values=close_prices.values
training_data_len=math.ceil(len(values)*0.8)

scaler=MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(values.reshape(-1,1))

train_data=scaled_data[0:training_data_len, :]

x_train=[]
y_train=[]

for i in range(60,len(train_data)):
    x_train.append(train_data[i-60:i,0])
    y_train.append(train_data[i,0])

x_train,y_train=np.array(x_train),np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0],x_train.shape[1],1))
```

In [18]:
```
test_data=scaled_data[training_data_len-60: , :]
x_test = []
y_test = values[training_data_len:]

for i in range(60,len(test_data)):
    x_test.append(test_data[i-60:i, 0])

x_test=np.array(x_test)
x_test=np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))
```

In [19]:
```
model=keras.Sequential()
model.add(layers.LSTM(100, return_sequences=True,input_shape=(x_train.shape[1], 1)))
model.add(layers.LSTM(100, return_sequences=False))
model.add(layers.Dense(25))
model.add(layers.Dense(1))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 100)           40800

 lstm_1 (LSTM)               (None, 100)               80400

 dense (Dense)               (None, 25)                2525

 dense_1 (Dense)             (None, 1)                 26

=================================================================
Total params: 123,751
Trainable params: 123,751
Non-trainable params: 0
_________________________________________________________________
```

In [20]:
```
model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(x_train,y_train,batch_size=1,epochs=3)
```

```
Epoch 1/3
1098/1098 [==============================] - 44s 33ms/step - loss: 8.5793e-04
Epoch 2/3
1098/1098 [==============================] - 36s 33ms/step - loss: 4.3776e-04
Epoch 3/3
1098/1098 [==============================] - 36s 33ms/step - loss: 3.1197e-04
```

Out[20]:
```
<keras.callbacks.History at 0x260433fb8e0>
```

In [21]:
```
predictions=model.predict(x_test)
predictions=scaler.inverse_transform(predictions)
rmse=np.sqrt(np.mean(predictions - y_test)**2)
rmse
```

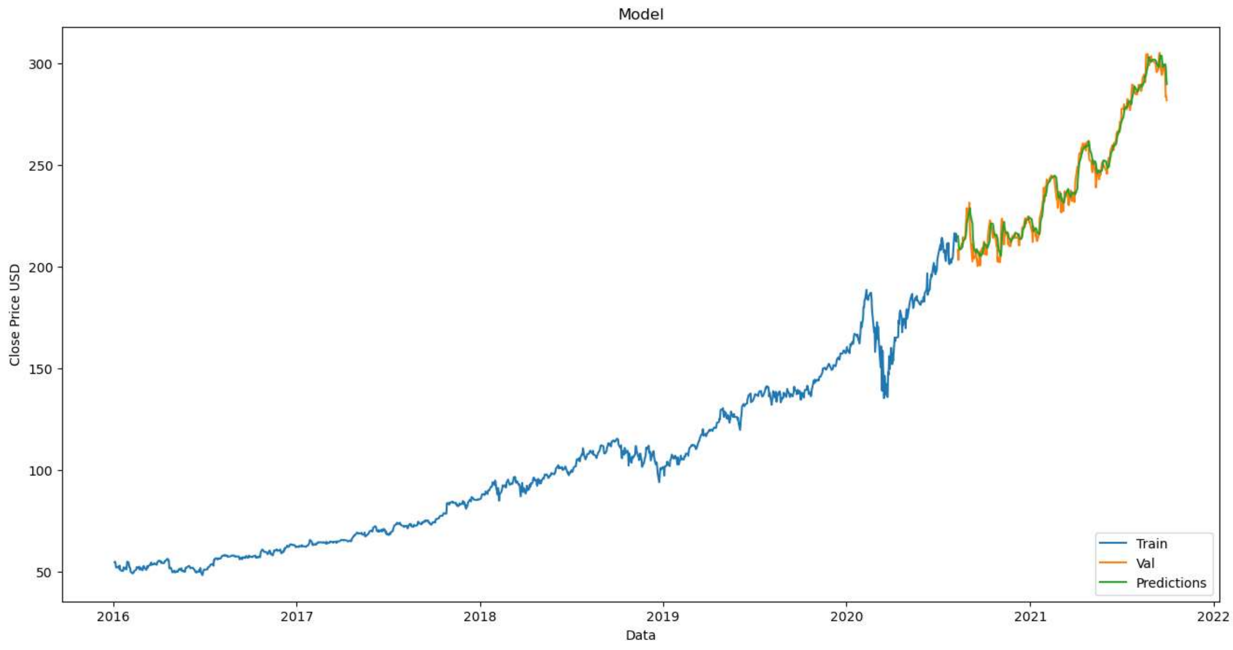
```
10/10 [==============================] - 2s 28ms/step
```

Out[21]:
```
0.6043666852799254
```

In [31]:
```
data=stock_data.filter(['Close'])
train=data[: training_data_len]
validation = data[training_data_len:]
validation['Predictions']=predictions
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Data')
plt.ylabel('Close Price USD')
plt.plot(train)
plt.plot(validation[['Close', 'Predictions']])
plt.legend(['Train','Val','Predictions'], loc='lower right')
plt.show()
```

```
C:\Users\Vishnu\AppData\Local\Temp\ipykernel_10536\1449559142.py:4: SettingWithCopyWa
rning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy
  validation['Predictions']=predictions
```



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```