SOURCE CODE

```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import LSTM, Dense
import gradio as gr

# Load data
df = pd.read_excel("Simulated_Stock_Data.xlsx")
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)

# EDA
plt.figure(figsize=(12,6))
sns.lineplot(data=df, x=df.index, y='Close')
plt.title('Stock Closing Prices Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.grid(True)
plt.show()

# Data Processing
```

```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df[['Close']])

sequence_length = 60
X, y = [], []
for i in range(sequence_length, len(scaled_data)):
    X.append(scaled_data[i-sequence_length:i, 0])
    y.append(scaled_data[i, 0])
X, y = np.array(X), np.array(y)
X = np.reshape(X, (X.shape[0], X.shape[1], 1))

# Model Building
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')

# Model Training
model.fit(X, y, epochs=10, batch_size=32)

# Model Evaluation
predicted = model.predict(X)
predicted_prices = scaler.inverse_transform(predicted.reshape(-1, 1))
actual_prices = scaler.inverse_transform(y.reshape(-1, 1))

plt.figure(figsize=(12,6))
plt.plot(actual_prices, label='Actual Price')
plt.plot(predicted_prices, label='Predicted Price')
plt.legend()
plt.title('Actual vs Predicted Stock Price')
plt.show()


mse = mean_squared_error(actual_prices, predicted_prices)
print(f"Mean Squared Error: {mse}")


# Gradio Interface


def predict_next_60_days():
    last_sequence = scaled_data[-sequence_length:]
    input_seq = last_sequence.reshape(1, sequence_length, 1)
    preds = []
    for _ in range(60):
        next_pred = model.predict(input_seq)[0][0]
        preds.append(next_pred)
        input_seq = np.append(input_seq[:, 1:, :], [[[next_pred]]], axis=1)
    return scaler.inverse_transform(np.array(preds).reshape(-1, 1)).flatten().tolist()


gr.Interface(fn=predict_next_60_days,
             inputs=[],
             outputs="plot",
             title="60-Day Stock Price Forecast").launch()
```