In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

In [ ]:
```
# Downloading stock data for Apple (AAPL)
data = yf.download('AAPL', start='2015-01-01', end='2022-12-31')
data = data[['Close']]  # using the closing price
data.head()
```

In [ ]:
```
plt.figure(figsize=(10, 4))
plt.plot(data['Close'])
plt.title("AAPL Stock Price")
plt.xlabel("Date")
plt.ylabel("Close Price")
plt.grid()
plt.show()
```

In [ ]:
```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)

X = []
y = []

time_step = 60  # Using 60 days to predict the 61st

for i in range(time_step, len(scaled_data)):
    X.append(scaled_data[i-time_step: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))
```

In [ ]:
```
train_size = int(len(X) * 0.8)

X_train = X[:train_size]
y_train = y[:train_size]
X_test = X[train_size:]
y_test = y[train_size:]
```

In [ ]:
```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
model.add(Dense(1))

model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

In [ ]:
```
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test)
```

In [ ]:
```
predicted = model.predict(X_test)
predicted = scaler.inverse_transform(predicted.reshape(-1, 1))
actual = scaler.inverse_transform(y_test.reshape(-1, 1))

plt.figure(figsize=(10, 4))
plt.plot(actual, label='Actual Price')
plt.plot(predicted, label='Predicted Price')
plt.title("AAPL Price Prediction")
plt.xlabel("Time")
plt.ylabel("Price")
plt.legend()
plt.grid()
plt.show()
```

In [ ]:
```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(actual, predicted))
print(f"Root Mean Squared Error: {rmse}")
```

In [ ]:
```

```