Implement a Stock price prediction model for predicting future stocks using historical data using Gated Recurrent Unit

```
import numpy as np
import pandas as pd
import yfinance as yf
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import GRU, Dense
import matplotlib.pyplot as plt
```

```
ticker_symbol = 'AAPL'
start_date = '2010-01-01'
end_date = '2022-01-01'
```

```
data = yf.download(ticker_symbol, start=start_date, end=end_date)
```

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

```
data
```

Date	Open	High	Low	Close	Adj Close	Volume
2010-01-04	7.622500	7.660714	7.585000	7.643214	6.470741	493729600
2010-01-05	7.664286	7.699643	7.616071	7.656429	6.481927	601904800
2010-01-06	7.656429	7.686786	7.526786	7.534643	6.378824	552160000
2010-01-07	7.562500	7.571429	7.466071	7.520714	6.367033	477131200
2010-01-08	7.510714	7.571429	7.466429	7.570714	6.409362	447610800
...	...	...	...	...	...	...
2021-12-27	177.089996	180.419998	177.070007	180.330002	178.065674	74919600
2021-12-28	180.160004	181.330002	178.529999	179.289993	177.038696	79144300
2021-12-29	179.330002	180.630005	178.139999	179.380005	177.127594	62348900
2021-12-30	179.470001	180.570007	178.089996	178.199997	175.962402	59773000
2021-12-31	178.089996	179.229996	177.259995	177.570007	175.340302	64062300

3021 rows × 6 columns

```
close_prices = data['Close'].values.reshape(-1, 1)
```

```
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(close_prices)
```

```
def create_dataset(data, time_step):
    X, y = [], []
    for i in range(len(data)-time_step-1):
        X.append(data[i:(i+time_step), 0])
        y.append(data[i + time_step, 0])
    return np.array(X), np.array(y)
```

```
time_step = 30
```

```
X, y = create_dataset(scaled_data, time_step)
```

```
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

```
model = Sequential()
model.add(GRU(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(GRU(units=50))
model.add(Dense(units=1))
```

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


model.fit(X_train, y_train, epochs=100, batch_size=64)
```

```
Epoch 1/100
38/38 [==============================] - 7s 42ms/step - loss: 8.0610e-04 - accuracy: 0.0000e+00
Epoch 2/100
38/38 [==============================] - 2s 41ms/step - loss: 2.2976e-05 - accuracy: 0.0000e+00
Epoch 3/100
38/38 [==============================] - 2s 40ms/step - loss: 1.7662e-05 - accuracy: 0.0000e+00
Epoch 4/100
38/38 [==============================] - 2s 40ms/step - loss: 1.7049e-05 - accuracy: 0.0000e+00
Epoch 5/100
38/38 [==============================] - 2s 41ms/step - loss: 1.6964e-05 - accuracy: 0.0000e+00
Epoch 6/100
38/38 [==============================] - 3s 67ms/step - loss: 1.6387e-05 - accuracy: 0.0000e+00
Epoch 7/100
38/38 [==============================] - 2s 49ms/step - loss: 1.6875e-05 - accuracy: 0.0000e+00
Epoch 8/100
38/38 [==============================] - 2s 58ms/step - loss: 1.6760e-05 - accuracy: 0.0000e+00
Epoch 9/100
38/38 [==============================] - 2s 40ms/step - loss: 1.7468e-05 - accuracy: 0.0000e+00
Epoch 10/100
38/38 [==============================] - 2s 41ms/step - loss: 1.5488e-05 - accuracy: 0.0000e+00
Epoch 11/100
38/38 [==============================] - 2s 40ms/step - loss: 1.5590e-05 - accuracy: 0.0000e+00
Epoch 12/100
38/38 [==============================] - 2s 40ms/step - loss: 1.4748e-05 - accuracy: 0.0000e+00
Epoch 13/100
38/38 [==============================] - 3s 67ms/step - loss: 1.4596e-05 - accuracy: 0.0000e+00
Epoch 14/100
38/38 [==============================] - 4s 92ms/step - loss: 1.6386e-05 - accuracy: 0.0000e+00
Epoch 15/100
38/38 [==============================] - 2s 51ms/step - loss: 1.4804e-05 - accuracy: 0.0000e+00
Epoch 16/100
38/38 [==============================] - 3s 67ms/step - loss: 1.3669e-05 - accuracy: 0.0000e+00
Epoch 17/100
38/38 [==============================] - 2s 56ms/step - loss: 1.3039e-05 - accuracy: 0.0000e+00
Epoch 18/100
38/38 [==============================] - 2s 41ms/step - loss: 1.3046e-05 - accuracy: 0.0000e+00
Epoch 19/100
38/38 [==============================] - 2s 40ms/step - loss: 1.2918e-05 - accuracy: 0.0000e+00
Epoch 20/100
38/38 [==============================] - 3s 68ms/step - loss: 1.3258e-05 - accuracy: 0.0000e+00
Epoch 21/100
38/38 [==============================] - 2s 52ms/step - loss: 1.3225e-05 - accuracy: 0.0000e+00
Epoch 22/100
38/38 [==============================] - 2s 42ms/step - loss: 1.2052e-05 - accuracy: 0.0000e+00
Epoch 23/100
38/38 [==============================] - 2s 40ms/step - loss: 1.1751e-05 - accuracy: 0.0000e+00
Epoch 24/100
38/38 [==============================] - 2s 40ms/step - loss: 1.2054e-05 - accuracy: 0.0000e+00
Epoch 25/100
38/38 [==============================] - 2s 41ms/step - loss: 1.1699e-05 - accuracy: 0.0000e+00
Epoch 26/100
38/38 [==============================] - 2s 40ms/step - loss: 1.1842e-05 - accuracy: 0.0000e+00
Epoch 27/100
38/38 [==============================] - 2s 45ms/step - loss: 1.2059e-05 - accuracy: 0.0000e+00
Epoch 28/100
38/38 [==============================] - 3s 67ms/step - loss: 1.0917e-05 - accuracy: 0.0000e+00
Epoch 29/100
38/38 [==============================] - 2s 40ms/step - loss: 1.1099e-05 - accuracy: 0.0000e+00
```

```
model.evaluate(X,y)
```

```
94/94 [==============================] - 2s 10ms/step - loss: 4.8620e-05 - accuracy: 3.3445e-04
[4.862000059802085e-05, 0.00033444815198890865]
```

```
predictions = model.predict(X_test)
predictions = scaler.inverse_transform(predictions)
y_test=scaler.inverse_transform(y_test.reshape(-1, 1))

plt.figure(figsize=(14, 7))
plt.plot(data.index[train_size + time_step + 1:], y_test, color='blue', label='Actual Stock Prices')
plt.plot(data.index[train_size + time_step + 1:], predictions, color='red', label='Predicted Stock Prices')
plt.title('Stock Price Prediction')
plt.xlabel('Date')
plt.ylabel('Price')
plt.legend()
plt.show()
```

```
19/19 [==============================] - 1s 10ms/step
```