In [1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
import matplotlib.pyplot as plt
```

In [2]:
```
tf.random.set_seed(42)
```

In [3]:
```
data = pd.read_csv("Housing.csv")
```

In [4]:
```
data = pd.get_dummies(data, columns=['mainroad', 'guestroom', 'basement', 'hotwaterhea
```

In [7]:
```
scaler = StandardScaler()
data[['area', 'bedrooms', 'bathrooms', 'stories', 'parking']] = scaler.fit_transform(d
```

In [8]:
```
X = data.drop('price', axis=1)
y = data['price']
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [12]:
```
def create_sequences(data, sequence_length):
    sequences = []
    for i in range(len(data) - sequence_length):
        sequences.append(data[i:i+sequence_length])
    return np.array(sequences)
```

In [13]:
```
sequence_length = 5

X_train_sequences = create_sequences(X_train.to_numpy(), sequence_length)
y_train_sequences = y_train[sequence_length:]
X_test_sequences = create_sequences(X_test.to_numpy(), sequence_length)
y_test_sequences = y_test[sequence_length:]
```

In [14]:
```
model = tf.keras.Sequential([
    tf.keras.layers.LSTM(50, activation='relu', input_shape=(sequence_length, X_train_
    tf.keras.layers.Dense(1)
])
```

In [26]:
```
y_pred = model.predict(X_test_sequences)
```

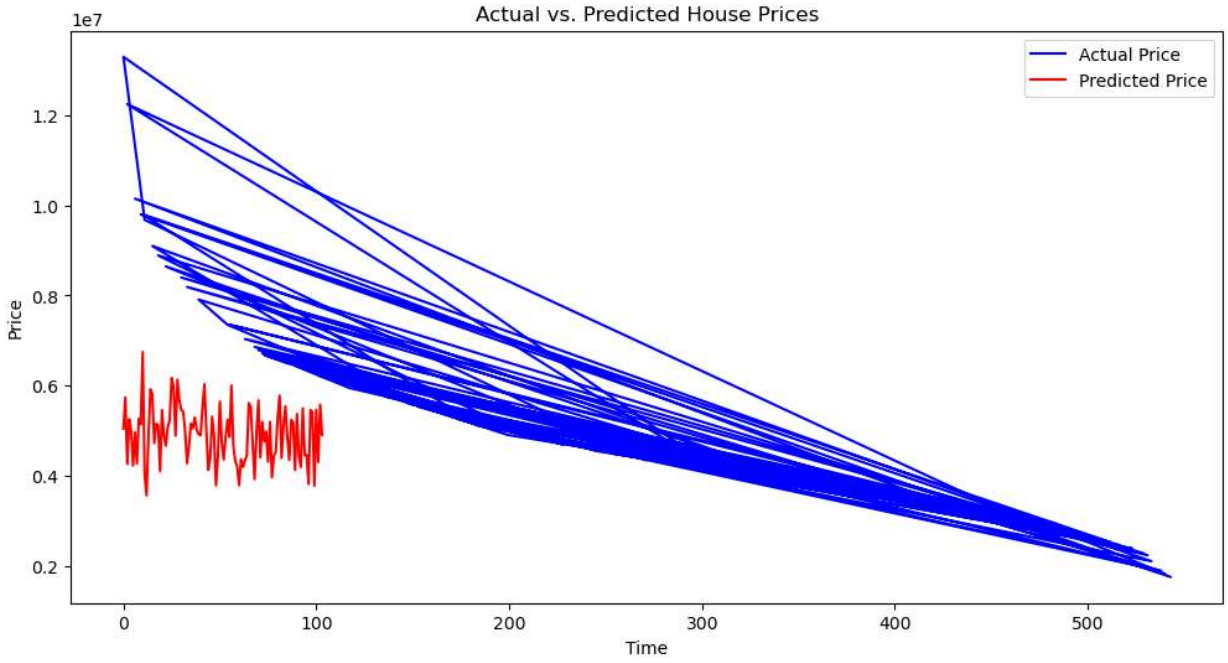
```
4/4 [==============================] - 0s 939us/step
```

In [27]:
```
mae = mean_absolute_error(y_test_sequences, y_pred)
mse = mean_squared_error(y_test_sequences, y_pred)
rmse = np.sqrt(mse)
```

In [28]:
```
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
```

```
MAE: 1893057.639423077
MSE: 5992412324034.577
RMSE: 2447940.4249357414
```

In [29]:
```
plt.figure(figsize=(12, 6))
plt.plot(y_test_sequences, label='Actual Price', color='blue')
plt.plot(y_pred, label='Predicted Price', color='red')
plt.title("Actual vs. Predicted House Prices")
plt.xlabel("Time")
plt.ylabel("Price")
plt.legend()
plt.show()
```



My Observation in creating this model is that tuning the epochs has helped in increasing the accuracy of the model but once I went above 500 epoch the model started to overfit and once I went below 300 the model is underfit. Hence 300 epoch is found to be best in my opinion.