In [25]:
```
# 1 - Packages
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Choose a plotting style
plt.style.use('ggplot')  # or use 'seaborn-v0_8' if available
```

In [29]:
```
# 2.1 - Load the Data
file_path = "housing.csv"  # Make sure this file is in the same directory as
data = pd.read_csv(file_path)

# Separate features and target
X = data[['size', 'location_score', 'bedrooms']].values  # Features
y = data['price'].values  # Target variable

# Check shapes
print("Shape of features (X):", X.shape)
print("Shape of target (y):", y.shape)
```

```
Shape of features (X): (500, 3)
Shape of target (y): (500,)
```

In [33]:
```
# 2.2 - Split the Data
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ranc

# Normalize (standardize) the feature values
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Confirm the shapes
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (400, 3)
X_test shape: (100, 3)
y_train shape: (400,)
y_test shape: (100,)
```

In [41]:
```
#3 - Neural Network Design
# Define the model using Input layer
model = Sequential([
    Input(shape=(3,)),              # Instead of input_dim=3
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1)  # Output layer for regression
])

# Compile the model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.01),
    loss='mean_squared_error',
    metrics=['mae']
)
```

In [43]:

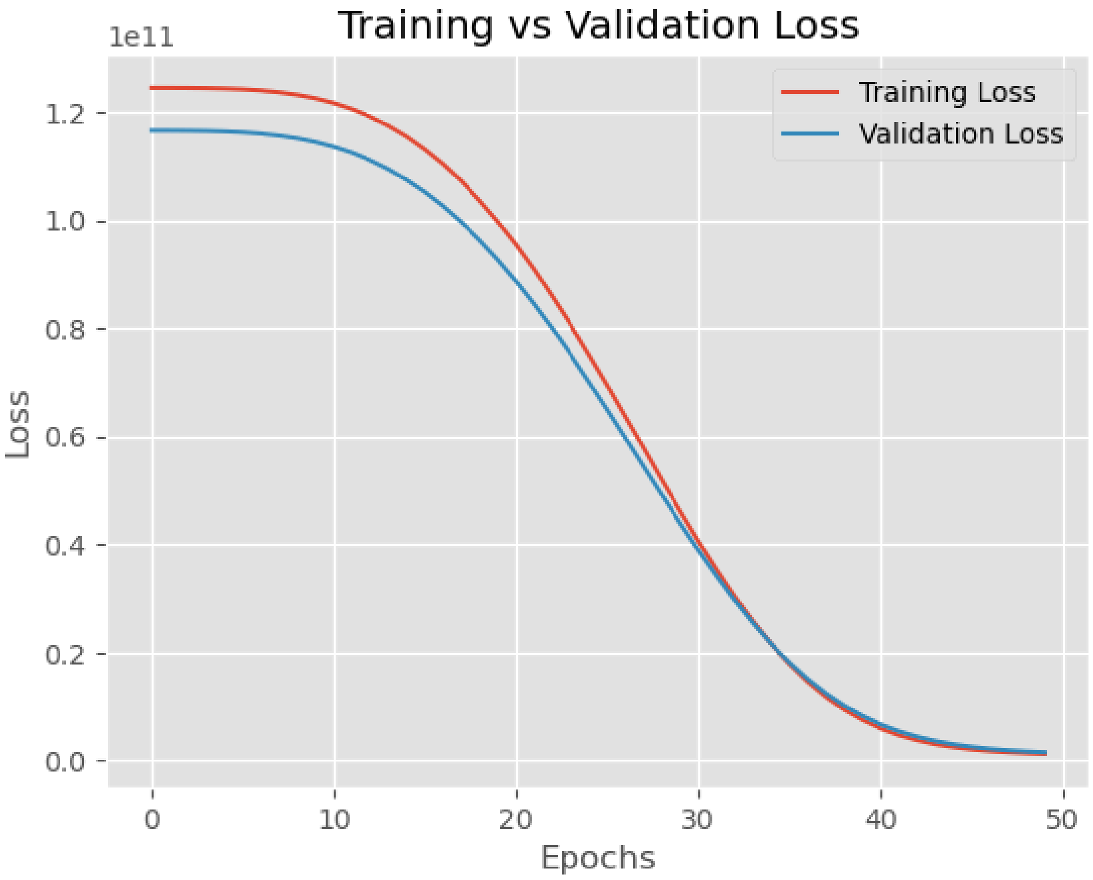
```
# Train the model
history = model.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=50,
    batch_size=32
)

# Plot training and validation loss
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training vs Validation Loss')
plt.show()
```

```
Epoch 1/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 125434011648.0000 - mae: 3490
67.0312 - val_loss: 116658511872.0000 - val_mae: 338453.4062
Epoch 2/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 125255245824.0000 - mae: 34928
7.6250 - val_loss: 116649476096.0000 - val_mae: 338440.0625
Epoch 3/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 124148211712.0000 - mae: 34763
7.8750 - val_loss: 116625940480.0000 - val_mae: 338405.4062
Epoch 4/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 127118557184.0000 - mae: 35151
0.9375 - val_loss: 116576501760.0000 - val_mae: 338332.8125
Epoch 5/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 125117947904.0000 - mae: 34891
2.0625 - val_loss: 116483489792.0000 - val_mae: 338196.3750
Epoch 6/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 124187942912.0000 - mae: 34737
0.0938 - val_loss: 116325851136.0000 - val_mae: 337965.6875
Epoch 7/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 123147001856.0000 - mae: 34620
3.1562 - val_loss: 116078829568.0000 - val_mae: 337604.7812
Epoch 8/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 122863149056.0000 - mae: 34565
4.8438 - val_loss: 115715473408.0000 - val_mae: 337073.8750
Epoch 9/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - loss: 121176309760.0000 - mae: 3435
74.0000 - val_loss: 115209191424.0000 - val_mae: 336333.6875
Epoch 10/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 124186238976.0000 - mae: 34742
1.0000 - val_loss: 114523914240.0000 - val_mae: 335331.9688
Epoch 11/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 120179720192.0000 - mae: 34182
1.7500 - val_loss: 113636646912.0000 - val_mae: 334029.0312
Epoch 12/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 119587282944.0000 - mae: 34149
9.8750 - val_loss: 112524787712.0000 - val_mae: 332392.3125
Epoch 13/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 119075725312.0000 - mae: 34022
7.2188 - val_loss: 111141363712.0000 - val_mae: 330346.6250
Epoch 14/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 116046225408.0000 - mae: 33600
6.7500 - val_loss: 109481246720.0000 - val_mae: 327874.8125
Epoch 15/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 119045120000.0000 - mae: 34021
3.3125 - val_loss: 107516936192.0000 - val_mae: 324931.5625
Epoch 16/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 113954086912.0000 - mae: 33272
4.4062 - val_loss: 105238806528.0000 - val_mae: 321478.8125
Epoch 17/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 111287885824.0000 - mae: 32914
0.1562 - val_loss: 102616997888.0000 - val_mae: 317463.9062
Epoch 18/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 108116508672.0000 - mae: 32466
7.8438 - val_loss: 99651043328.0000 - val_mae: 312868.0625
Epoch 19/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 101109727232.0000 - mae: 31419
8.6875 - val_loss: 96360366080.0000 - val_mae: 307675.5000
```

```
Epoch 20/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 98997805056.0000 - mae: 31075
7.2188 - val_loss: 92725272576.0000 - val_mae: 301839.7812
Epoch 21/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 95637979136.0000 - mae: 30539
1.5938 - val_loss: 88752046080.0000 - val_mae: 295334.5625
Epoch 22/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 93062660096.0000 - mae: 30115
4.4062 - val_loss: 84466393088.0000 - val_mae: 288151.4375
Epoch 23/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 85685903360.0000 - mae: 28935
6.6875 - val_loss: 79935635456.0000 - val_mae: 280335.3438
Epoch 24/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 81754415104.0000 - mae: 28238
1.3750 - val_loss: 75127980032.0000 - val_mae: 271806.3125
Epoch 25/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 75998380032.0000 - mae: 27253
1.3125 - val_loss: 70105604096.0000 - val_mae: 262587.6250
Epoch 26/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 72728920064.0000 - mae: 26635
5.6875 - val_loss: 64945807360.0000 - val_mae: 252756.5312
Epoch 27/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 63840653312.0000 - mae: 24973
2.5156 - val_loss: 59685797888.0000 - val_mae: 242290.2031
Epoch 28/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 59744395264.0000 - mae: 24182
4.3438 - val_loss: 54381309952.0000 - val_mae: 231251.0938
Epoch 29/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 53104996352.0000 - mae: 22818
3.1562 - val_loss: 49113325568.0000 - val_mae: 219705.7031
Epoch 30/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 47871143936.0000 - mae: 21652
9.0312 - val_loss: 43934801920.0000 - val_mae: 207707.2344
Epoch 31/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 41698549760.0000 - mae: 20221
3.3125 - val_loss: 38937550848.0000 - val_mae: 195374.3281
Epoch 32/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 35676721152.0000 - mae: 18698
1.4844 - val_loss: 34143350784.0000 - val_mae: 182724.7188
Epoch 33/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 30821648384.0000 - mae: 17350
7.5781 - val_loss: 29602574336.0000 - val_mae: 169849.5781
Epoch 34/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 25948841984.0000 - mae: 15876
6.5000 - val_loss: 25389721600.0000 - val_mae: 156915.4844
Epoch 35/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 22396825600.0000 - mae: 14722
0.1250 - val_loss: 21529593856.0000 - val_mae: 144019.7500
Epoch 36/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 18258094080.0000 - mae: 13229
9.3750 - val_loss: 18077585408.0000 - val_mae: 131363.2969
Epoch 37/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 15136901120.0000 - mae: 11976
0.4688 - val_loss: 15025245184.0000 - val_mae: 119031.7031
Epoch 38/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 12022173696.0000 - mae: 10575
5.3281 - val_loss: 12379179008.0000 - val_mae: 107160.5469
```

```
Epoch 39/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 9557672960.0000 - mae: 93011.1
641 - val_loss: 10117072896.0000 - val_mae: 95857.2188
Epoch 40/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 7911838720.0000 - mae: 83215.5
156 - val_loss: 8213830144.0000 - val_mae: 85195.9219
Epoch 41/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 6241523200.0000 - mae: 73132.0
078 - val_loss: 6656406528.0000 - val_mae: 75467.0781
Epoch 42/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 4856010240.0000 - mae: 62107.1
602 - val_loss: 5401441792.0000 - val_mae: 66946.5938
Epoch 43/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3899867392.0000 - mae: 55026.1
406 - val_loss: 4389574656.0000 - val_mae: 59318.0078
Epoch 44/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 3030742784.0000 - mae: 48029.5
977 - val_loss: 3596845056.0000 - val_mae: 52786.1016
Epoch 45/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 2535399424.0000 - mae: 43661.2
812 - val_loss: 2980361728.0000 - val_mae: 47379.8828
Epoch 46/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1997183104.0000 - mae: 37788.5
625 - val_loss: 2507908352.0000 - val_mae: 43085.4062
Epoch 47/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1693281280.0000 - mae: 35069.0
469 - val_loss: 2151109120.0000 - val_mae: 39704.9766
Epoch 48/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1614479232.0000 - mae: 33947.3
516 - val_loss: 1876070400.0000 - val_mae: 37009.3633
Epoch 49/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1388614912.0000 - mae: 31609.5
293 - val_loss: 1673164416.0000 - val_mae: 34727.6094
Epoch 50/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1157646848.0000 - mae: 28320.2
266 - val_loss: 1519882496.0000 - val_mae: 32793.8242
```



In [45]:

```
# Evaluate the model on the test set
loss, mae = model.evaluate(X_test, y_test)

print(f"Test Loss: {loss}")
print(f"Test Mean Absolute Error (MAE): {mae}")
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 1251984768.0000 - mae: 27745.771
5
Test Loss: 1242353280.0
Test Mean Absolute Error (MAE): 27912.810546875
```

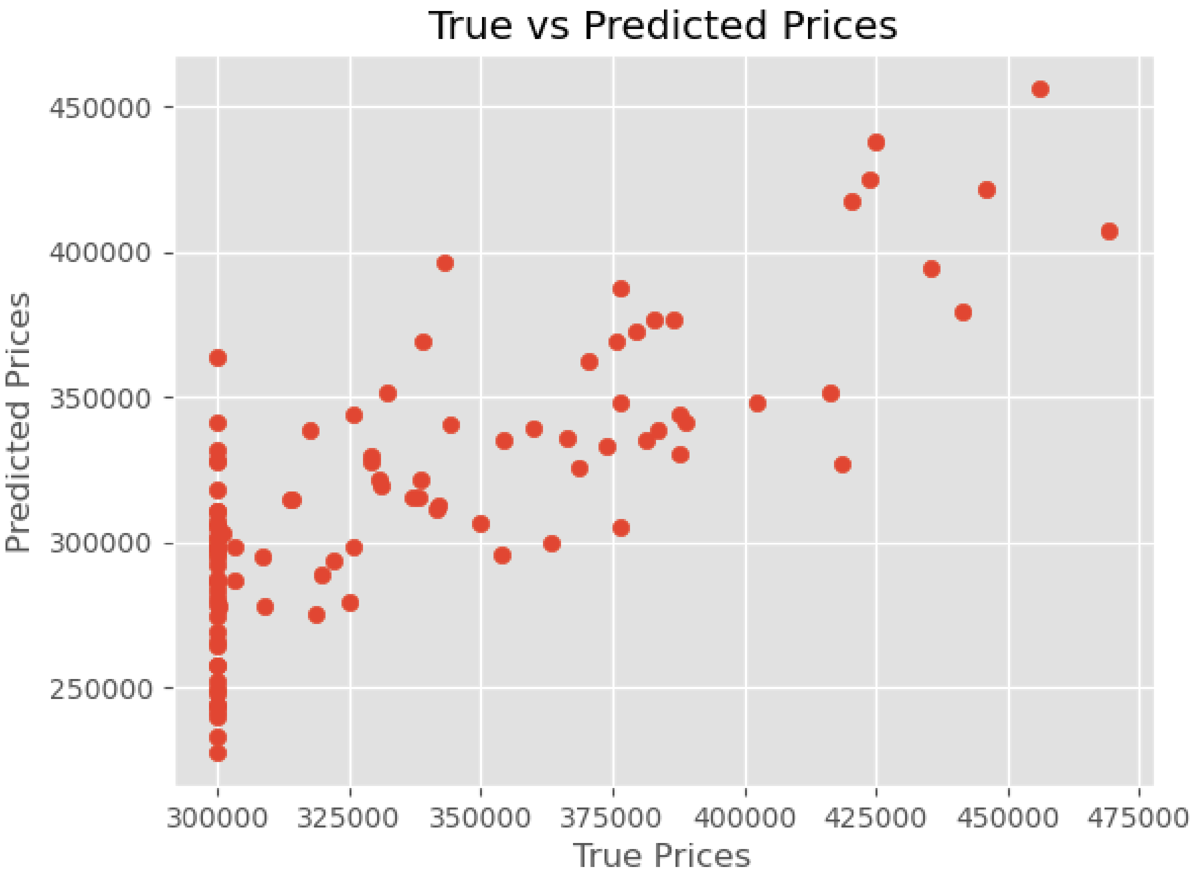
In [47]:

```
# 5.1 - Make Predictions
predictions = model.predict(X_test)
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
```

In [49]:

```
# 5.2 - Visualize Predictions
plt.scatter(y_test, predictions)
plt.xlabel('True Prices')
plt.ylabel('Predicted Prices')
plt.title('True vs Predicted Prices')
plt.show()
```


True vs Predicted Prices
Predicted Prices
450000
400000
350000
300000
250000
True Prices
300000
325000
350000
375000
400000
425000
450000
475000

```
In [ ]:
```