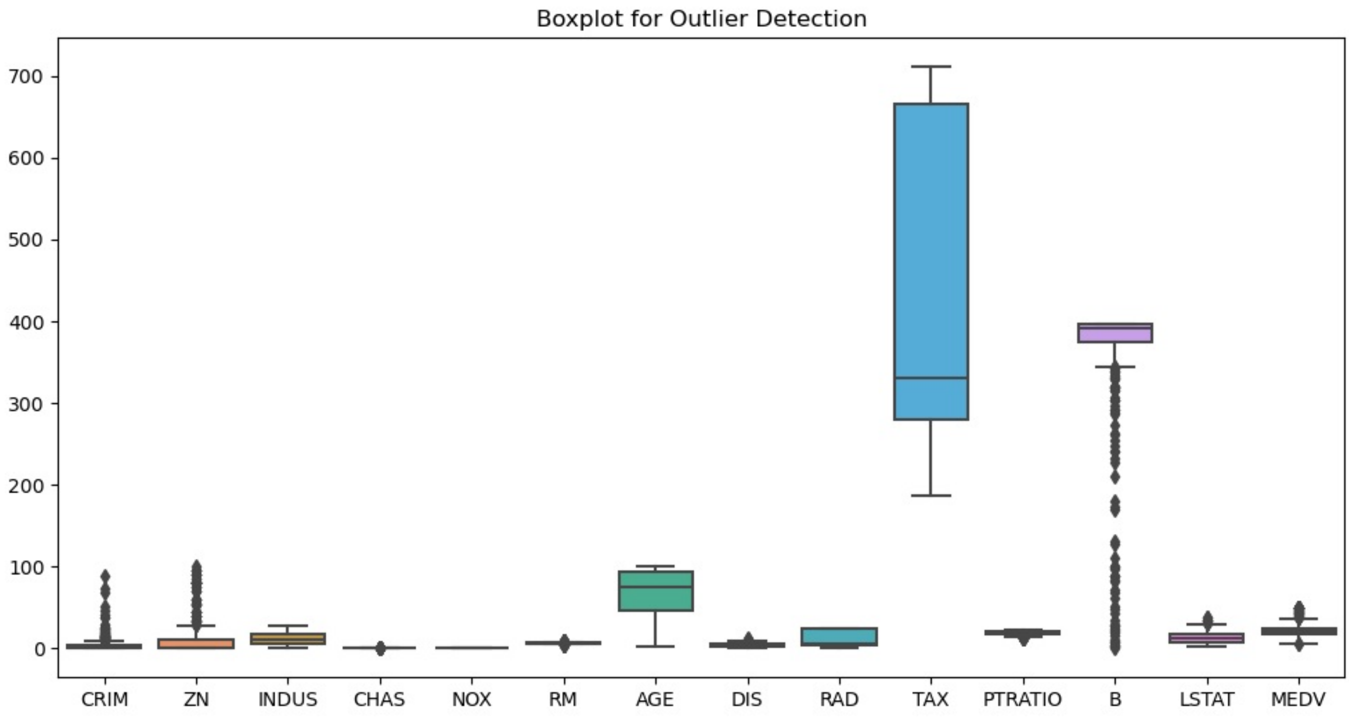
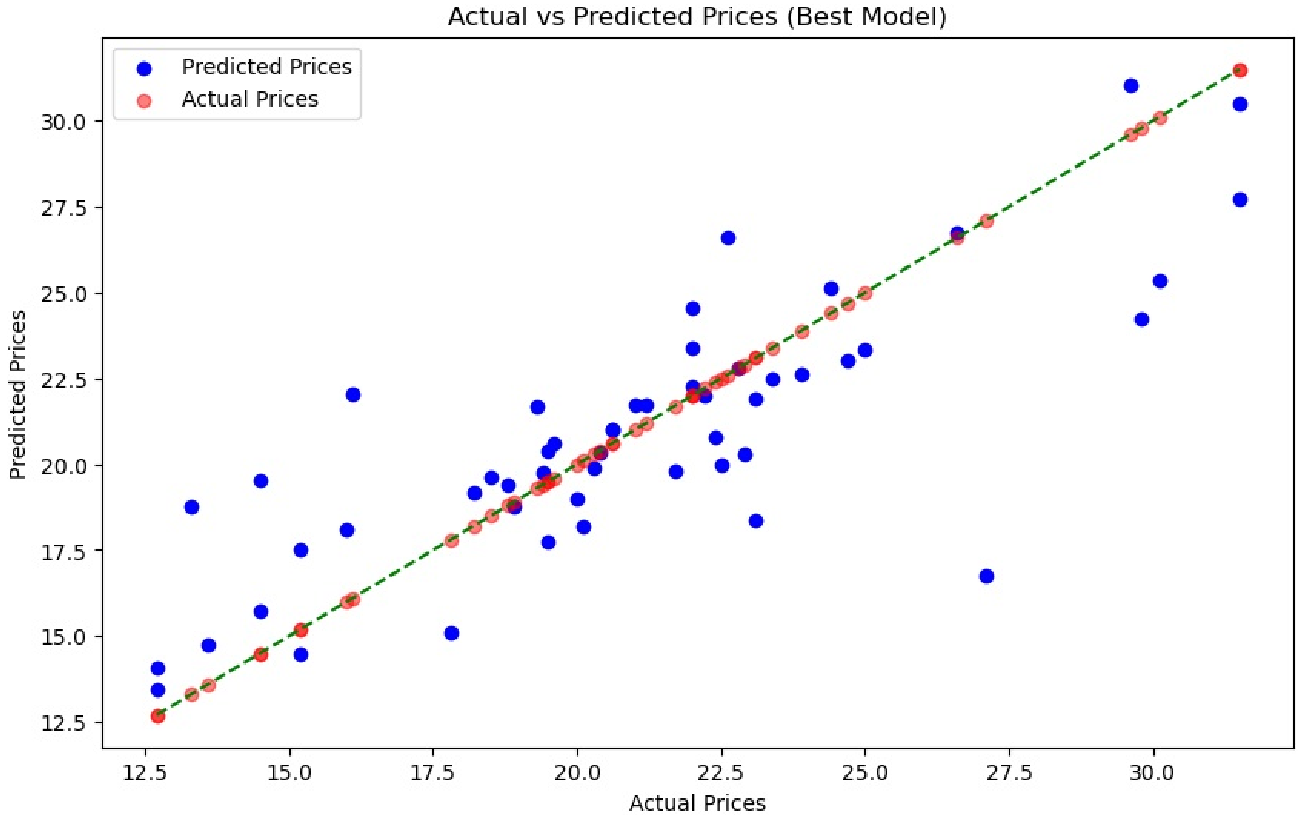
In [3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

# Load the dataset
data = pd.read_csv('HousingData.csv')  # Replace with your dataset path

# Display the first few rows of the dataset
print(data.head())

# Data Preprocessing
# Check for missing values
print("Missing values in each column:")
print(data.isnull().sum())

# Handle missing values (if any)
data.fillna(data.mean(), inplace=True)  # Filling missing values with mean

# Check for outliers using boxplots
plt.figure(figsize=(12, 6))
sns.boxplot(data=data)
plt.title('Boxplot for Outlier Detection')
plt.show()

# Remove outliers using IQR method
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)
IQR = Q3 - Q1
data = data[~((data < (Q1 - 1.5 * IQR)) | (data > (Q3 + 1.5 * IQR))).any(axis=1)]

# Feature Selection
# Assuming 'MEDV' is the target variable (house prices)
X = data.drop('MEDV', axis=1)  # Features
y = data['MEDV']  # Target variable

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling (optional but recommended for some models)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)  # Fit and transform training data
X_test = scaler.transform(X_test)  # Transform testing data

# Model Selection
# Initialize models
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(),
    'Gradient Boosting': GradientBoostingRegressor()
}

# Step F: Training and Evaluation
for model_name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Evaluate the model
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'{model_name} - Mean Squared Error: {mse:.2f}, R-squared: {r2:.2f}')

    # Cross-validation scores
    cv_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')
    print(f'{model_name} - Cross-validation MSE: {cv_scores.mean():.2f}')

# Fine-tuning using GridSearchCV on the Gradient Boosting model
param_grid = {
    'n_estimators': [100, 200],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7]
```

```
}

grid_search = GridSearchCV(GradientBoostingRegressor(), param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

# Best model from grid search
best_model = grid_search.best_estimator_
y_pred_best = best_model.predict(X_test)

# Evaluate the best model
best_mse = mean_squared_error(y_test, y_pred_best)
best_r2 = r2_score(y_test, y_pred_best)

print(f'Best Gradient Boosting Model - Mean Squared Error: {best_mse:.2f}, R-squared: {best_r2:.2f}')

# Visualization of Actual vs Predicted Prices for the Best Model
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_best, color='blue', label='Predicted Prices')
plt.scatter(y_test, y_test, color='red', label='Actual Prices', alpha=0.5)
plt.xlabel('Actual Prices')
plt.ylabel('Predicted Prices')
plt.title('Actual vs Predicted Prices (Best Model)')
plt.legend()
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='green', linestyle='--')
plt.show()
```

```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31   0.0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07   0.0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07   0.0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18   0.0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18   0.0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90    NaN  36.2
Missing values in each column:
CRIM       20
ZN         20
INDUS      20
CHAS       20
NOX         0
RM          0
AGE        20
DIS         0
RAD         0
TAX         0
PTRATIO     0
B           0
LSTAT      20
MEDV        0
dtype: int64
```



```
Linear Regression - Mean Squared Error: 7.81, R-squared: 0.63
Linear Regression - Cross-validation MSE: -14.06
Decision Tree - Mean Squared Error: 7.75, R-squared: 0.64
Decision Tree - Cross-validation MSE: -25.03
Gradient Boosting - Mean Squared Error: 6.81, R-squared: 0.68
Gradient Boosting - Cross-validation MSE: -11.90
Best Gradient Boosting Model - Mean Squared Error: 7.46, R-squared: 0.65
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js