In [15]:
```
!pip install pandas
import pandas as pd

# Load the two datasets (update with your actual paths)
data_mat = pd.read_csv('/Users/HarryXiao/Desktop/student-mat.csv', sep=';
data_por = pd.read_csv('/Users/HarryXiao/Desktop/student-por.csv', sep=';


# Select independent features for the model
selected_features = ['failures', 'studytime', 'higher', 'famsup', 'school
                     'Medu', 'Fedu', 'absences', 'health', 'activities',

# Prepare feature matrix (X) and target variable (y)
X = data_mat[selected_features]
y = data_mat['G3']

# Convert categorical variables to numeric (e.g., Yes/No to 1/0)
X = pd.get_dummies(X, drop_first=True)
```

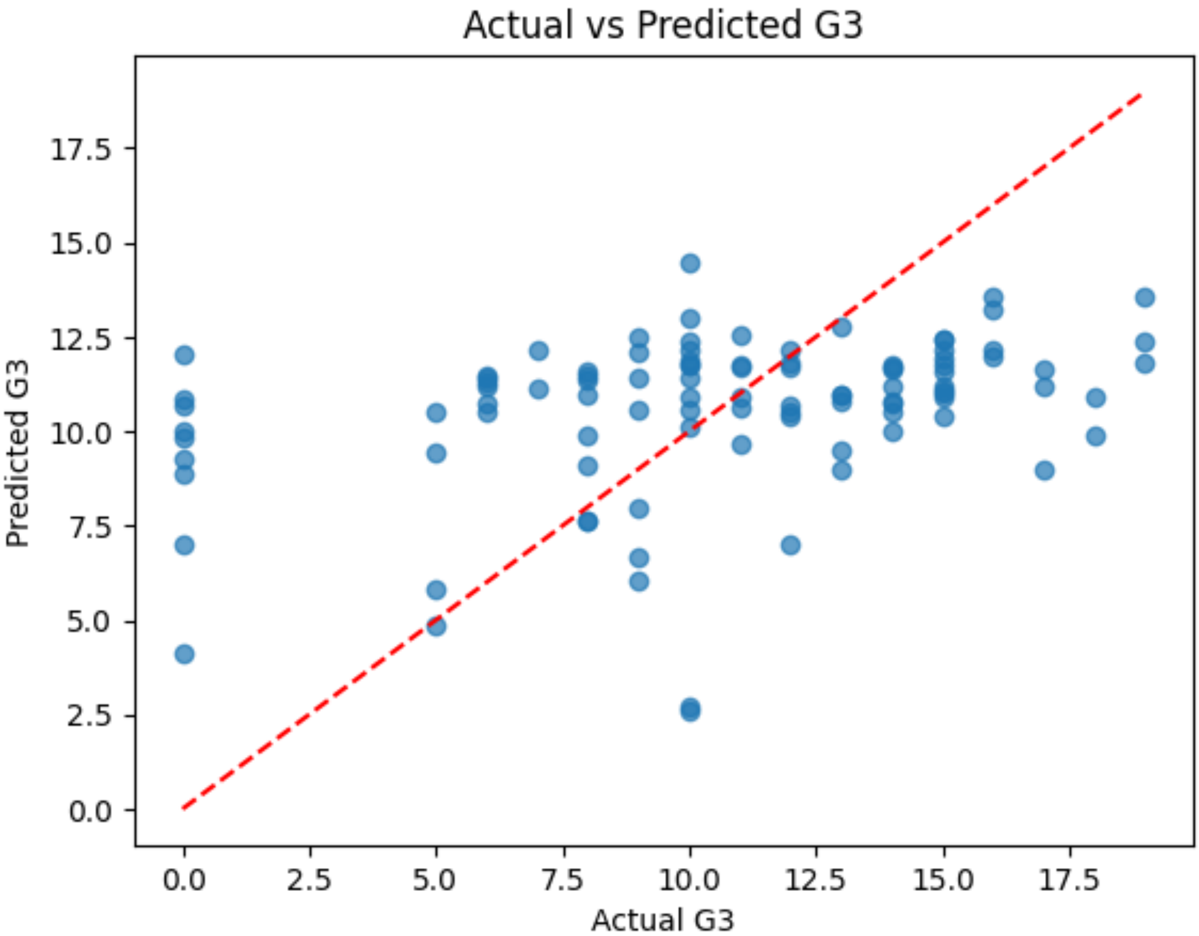
```
Requirement already satisfied: pandas in /Library/Frameworks/Python.framew
ork/Versions/3.13/lib/python3.13/site-packages (2.2.3)
Requirement already satisfied: numpy>=1.26.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.13/lib/python3.13/site-packages (from pandas) (2.1.
3)
Requirement already satisfied: python-dateutil>=2.8.2 in /Library/Framewor
ks/Python.framework/Versions/3.13/lib/python3.13/site-packages (from panda
s) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /Library/Frameworks/Python.
framework/Versions/3.13/lib/python3.13/site-packages (from pandas) (2024.
2)
Requirement already satisfied: tzdata>=2022.7 in /Library/Frameworks/Pytho
n.framework/Versions/3.13/lib/python3.13/site-packages (from pandas) (202
4.2)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.fram
ework/Versions/3.13/lib/python3.13/site-packages (from python-dateutil>=2.
8.2->pandas) (1.16.0)
```

In [12]:
```
!pip install scikit-learn
!pip install matplotlib

from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,

print("Training set size:", X_train.shape[0])
print("Testing set size:", X_test.shape[0])
from sklearn.linear_model import LinearRegression

# Initialize the Linear Regression model
model = LinearRegression()
```

```
# Train the model
model.fit(X_train, y_train)

# Display coefficients
print("Model coefficients:", model.coef_)
print("Model intercept:", model.intercept_)
from sklearn.metrics import mean_squared_error, r2_score

# Make predictions on the test set
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"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R^2): {r2:.2f}")
import matplotlib.pyplot as plt

# Plot actual vs predicted values
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], colo
plt.xlabel('Actual G3')
plt.ylabel('Predicted G3')
plt.title('Actual vs Predicted G3')
plt.show()
```

```
Requirement already satisfied: scikit-learn in /Library/Frameworks/Python.
framework/Versions/3.13/lib/python3.13/site-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /Library/Frameworks/Pytho
n.framework/Versions/3.13/lib/python3.13/site-packages (from scikit-learn)
(2.1.3)
Requirement already satisfied: scipy>=1.6.0 in /Library/Frameworks/Python.
framework/Versions/3.13/lib/python3.13/site-packages (from scikit-learn) (
1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.13/lib/python3.13/site-packages (from scikit-learn)
(1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /Library/Framework
s/Python.framework/Versions/3.13/lib/python3.13/site-packages (from scikit
-learn) (3.5.0)
Requirement already satisfied: matplotlib in /Library/Frameworks/Python.fr
amework/Versions/3.13/lib/python3.13/site-packages (3.9.2)
Requirement already satisfied: contourpy>=1.0.1 in /Library/Frameworks/Pyt
hon.framework/Versions/3.13/lib/python3.13/site-packages (from matplotlib)
(1.3.1)
Requirement already satisfied: cycler>=0.10 in /Library/Frameworks/Python.
framework/Versions/3.13/lib/python3.13/site-packages (from matplotlib) (0.
12.1)
Requirement already satisfied: fonttools>=4.22.0 in /Library/Frameworks/Py
thon.framework/Versions/3.13/lib/python3.13/site-packages (from matplotli
b) (4.55.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /Library/Frameworks/Py
thon.framework/Versions/3.13/lib/python3.13/site-packages (from matplotli
```

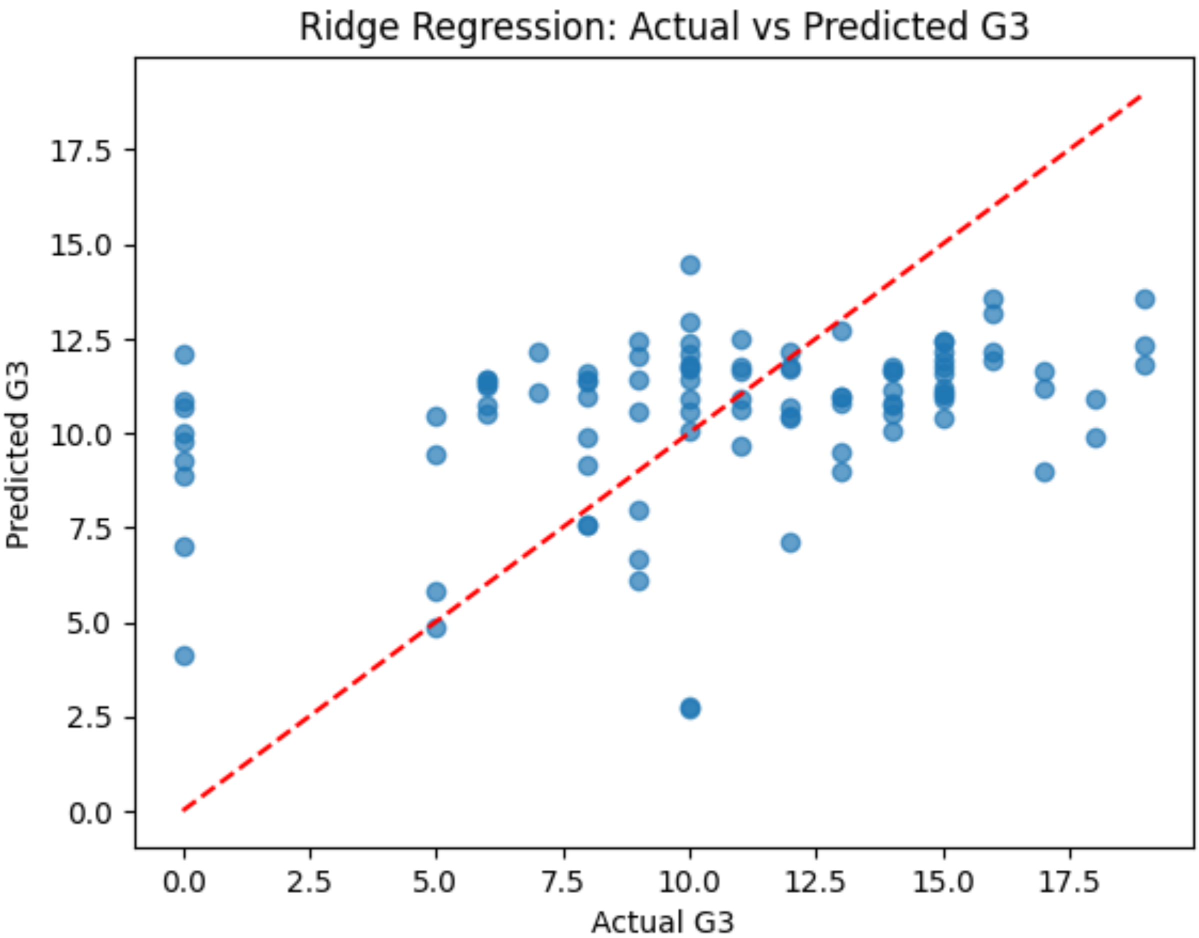
```
b) (1.4.7)
Requirement already satisfied: numpy>=1.23 in /Library/Frameworks/Python.f
ramework/Versions/3.13/lib/python3.13/site-packages (from matplotlib) (2.
1.3)
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Pyth
on.framework/Versions/3.13/lib/python3.13/site-packages (from matplotlib)
(24.1)
Requirement already satisfied: pillow>=8 in /Library/Frameworks/Python.fra
mework/Versions/3.13/lib/python3.13/site-packages (from matplotlib) (11.0.
0)
Requirement already satisfied: pyparsing>=2.3.1 in /Library/Frameworks/Pyt
hon.framework/Versions/3.13/lib/python3.13/site-packages (from matplotlib)
(3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /Library/Framework
s/Python.framework/Versions/3.13/lib/python3.13/site-packages (from matplo
tlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /Library/Frameworks/Python.fram
ework/Versions/3.13/lib/python3.13/site-packages (from python-dateutil>=2.
7->matplotlib) (1.16.0)
Training set size: 296
Testing set size: 99
Model coefficients: [-1.83606119  0.31078281  0.65227662 -0.39917592  0.02
954406 -0.02392766
 -0.40042748  2.53897162 -1.07807732 -0.69114182 -0.42755101  0.98430227]
Model intercept: 8.527617876701644
Mean Squared Error (MSE): 20.32
R-squared (R^2): 0.12
```



In [ ]:
```
###The MSE was 20.32
###The R² value was 0.12, indicating that only 12% of the variance in the
###The scatter plot compares actual G3 scores with predicted ones. The re
```

In [13]:
```
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
alphas = [0.01, 0.1, 1, 10, 100]  # Test different alpha values
for alpha in alphas:
    ridge = Ridge(alpha=alpha)  # Initialize Ridge Regression with curren
    ridge.fit(X_train, y_train)  # Fit the model

    # Make predictions
    y_pred = ridge.predict(X_test)

    # Evaluate the model
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"Alpha: {alpha}")
    print(f"Mean Squared Error: {mse:.2f}")
    print(f"R-squared: {r2:.2f}\n")
best_alpha = 1  # Replace with the chosen alpha
ridge_best = Ridge(alpha=best_alpha)
ridge_best.fit(X_train, y_train)

y_pred_best = ridge_best.predict(X_test)
mse_best = mean_squared_error(y_test, y_pred_best)
r2_best = r2_score(y_test, y_pred_best)
print(f"Best Alpha: {best_alpha}")
print(f"Final Mean Squared Error: {mse_best:.2f}")
print(f"Final R-squared: {r2_best:.2f}")
import matplotlib.pyplot as plt

plt.scatter(y_test, y_pred_best, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], colo
plt.xlabel("Actual G3")
plt.ylabel("Predicted G3")
plt.title("Ridge Regression: Actual vs Predicted G3")
plt.show()
```

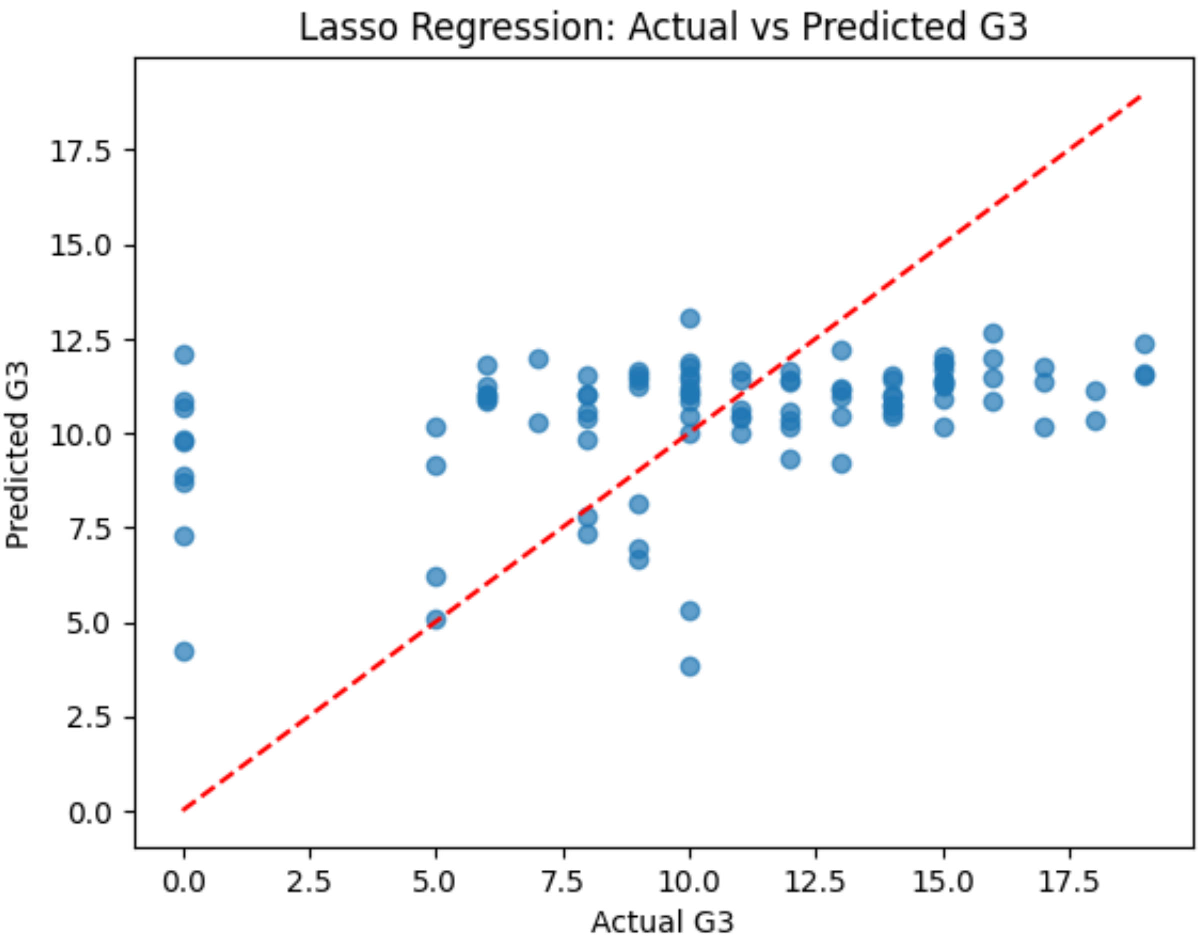
```
Alpha: 0.01
Mean Squared Error: 20.32
R-squared: 0.12

Alpha: 0.1
Mean Squared Error: 20.31
R-squared: 0.12

Alpha: 1
Mean Squared Error: 20.25
R-squared: 0.12

Alpha: 10
Mean Squared Error: 19.86
R-squared: 0.14

Alpha: 100
Mean Squared Error: 19.52
R-squared: 0.15

Best Alpha: 1
Final Mean Squared Error: 20.25
Final R-squared: 0.12
```



In [ ]:
```
###As alpha increased, the Mean Squared Error (MSE) slightly decreased
###The R-squared (R²) improved slightly to 0.15 as alpha increased, but i
```

In [14]:
```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import matplotlib.pyplot as plt

# Set different alpha values to test
alphas = [0.01, 0.1, 1, 10, 100]
results = []

# Iterate over each alpha to train and evaluate the model
for alpha in alphas:
    # Initialize the Lasso regression model
    model = Lasso(alpha=alpha, random_state=42)
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    # Calculate Mean Squared Error (MSE) and R-squared (R²)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results.append((alpha, mse, r2))

    # Print the evaluation metrics for each alpha
    print(f"Alpha: {alpha}")
    print(f"Mean Squared Error: {mse:.2f}")
    print(f"R-squared: {r2:.2f}\n")

# Identify the best alpha based on the minimum MSE
best_result = min(results, key=lambda x: x[1])
best_alpha = best_result[0]

# Print the best alpha and corresponding performance
print(f"Best Alpha: {best_alpha}")
print(f"Final Mean Squared Error: {best_result[1]:.2f}")
print(f"Final R-squared: {best_result[2]:.2f}")

# Retrain the model with the best alpha and visualize results
model = Lasso(alpha=best_alpha, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Plot actual vs predicted values
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], colo
plt.xlabel("Actual G3")
plt.ylabel("Predicted G3")
plt.title("Lasso Regression: Actual vs Predicted G3")
plt.show()
```

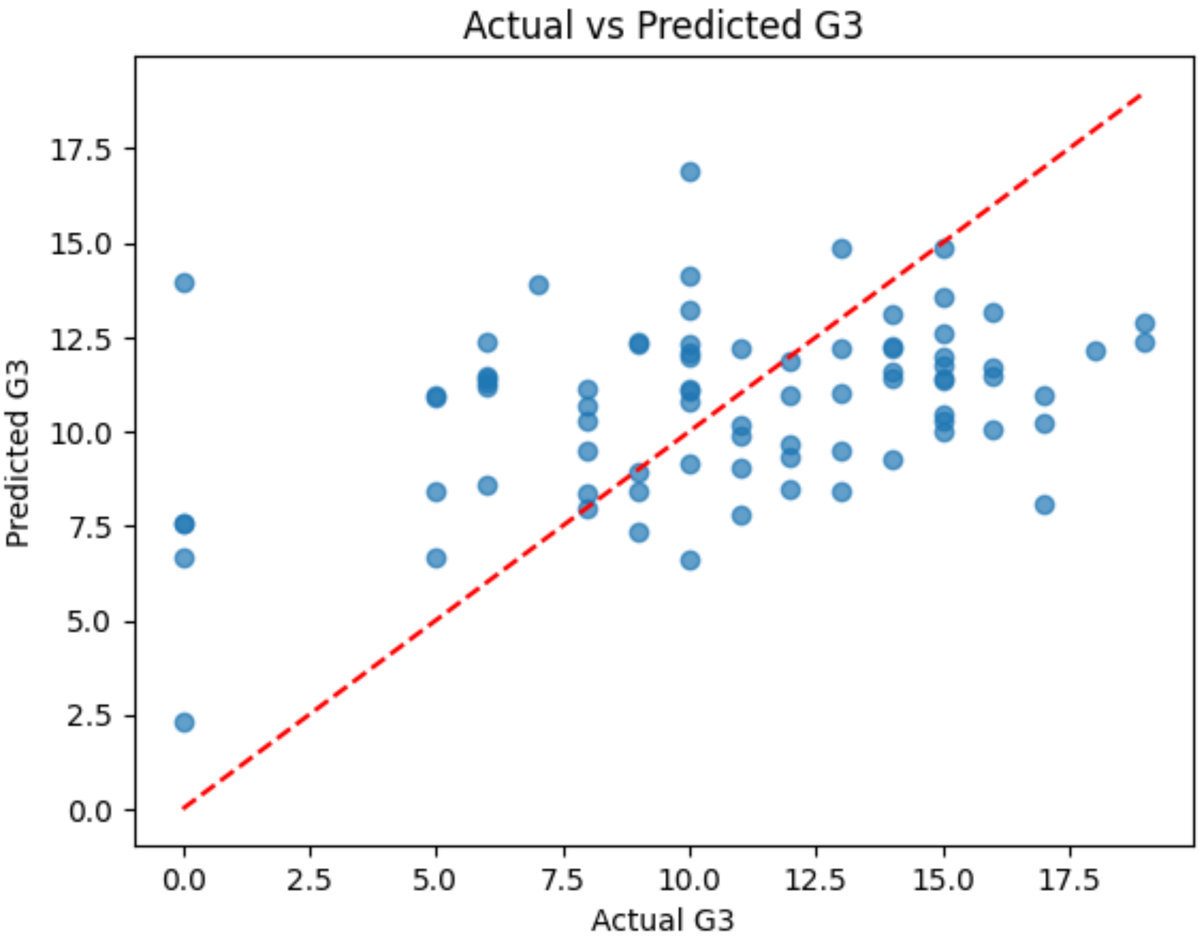
```
Alpha: 0.01
Mean Squared Error: 20.18
R-squared: 0.12

Alpha: 0.1
Mean Squared Error: 19.37
R-squared: 0.16

Alpha: 1
Mean Squared Error: 22.25
R-squared: 0.03

Alpha: 10
Mean Squared Error: 23.05
R-squared: -0.00

Alpha: 100
Mean Squared Error: 23.05
R-squared: -0.00

Best Alpha: 0.1
Final Mean Squared Error: 19.37
Final R-squared: 0.16
```

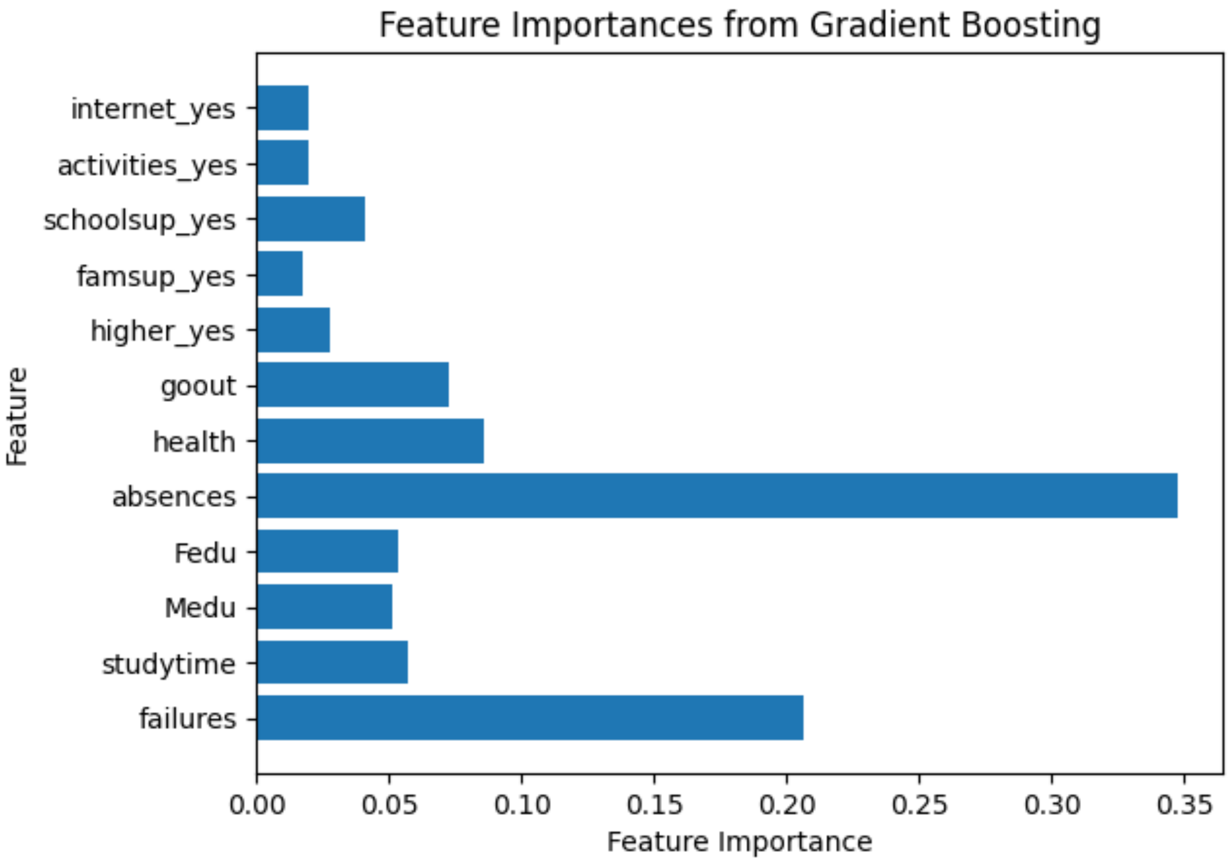


In [ ]:
```
###The best alpha value is 0.1, which minimizes the Mean Squared Error (M
```

In [2]:
```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset (replace with actual dataset path)
data = pd.read_csv('student-mat.csv', sep=';')

# Select features and target
selected_features = ['failures', 'studytime', 'higher', 'famsup', 'school
                     'Medu', 'Fedu', 'absences', 'health', 'activities',
X = data[selected_features]
y = data['G3']

# Convert categorical variables to numeric (if necessary)
X = pd.get_dummies(X, drop_first=True)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

# Initialize and train the Gradient Boosting Regressor
model = GradientBoostingRegressor(random_state=42)
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"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R^2): {r2:.2f}")

# Plot actual vs predicted values
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], colo
plt.xlabel('Actual G3')
plt.ylabel('Predicted G3')
plt.title('Actual vs Predicted G3')
plt.show()

# Plot feature importances
importances = model.feature_importances_
plt.barh(range(len(importances)), importances, tick_label=X.columns)
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importances from Gradient Boosting')
plt.show()
```

```
Mean Squared Error (MSE): 17.93
R-squared (R^2): 0.13
```

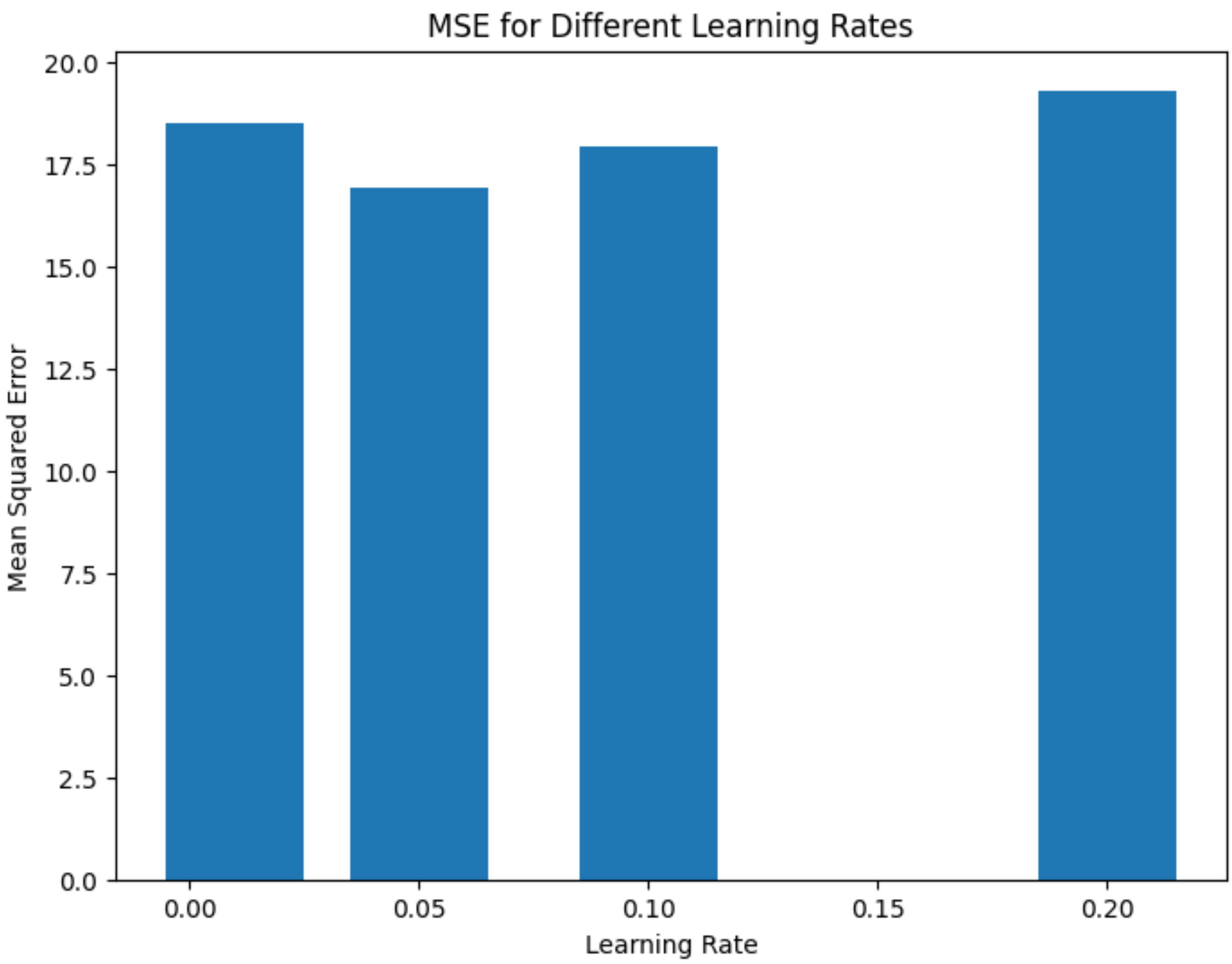


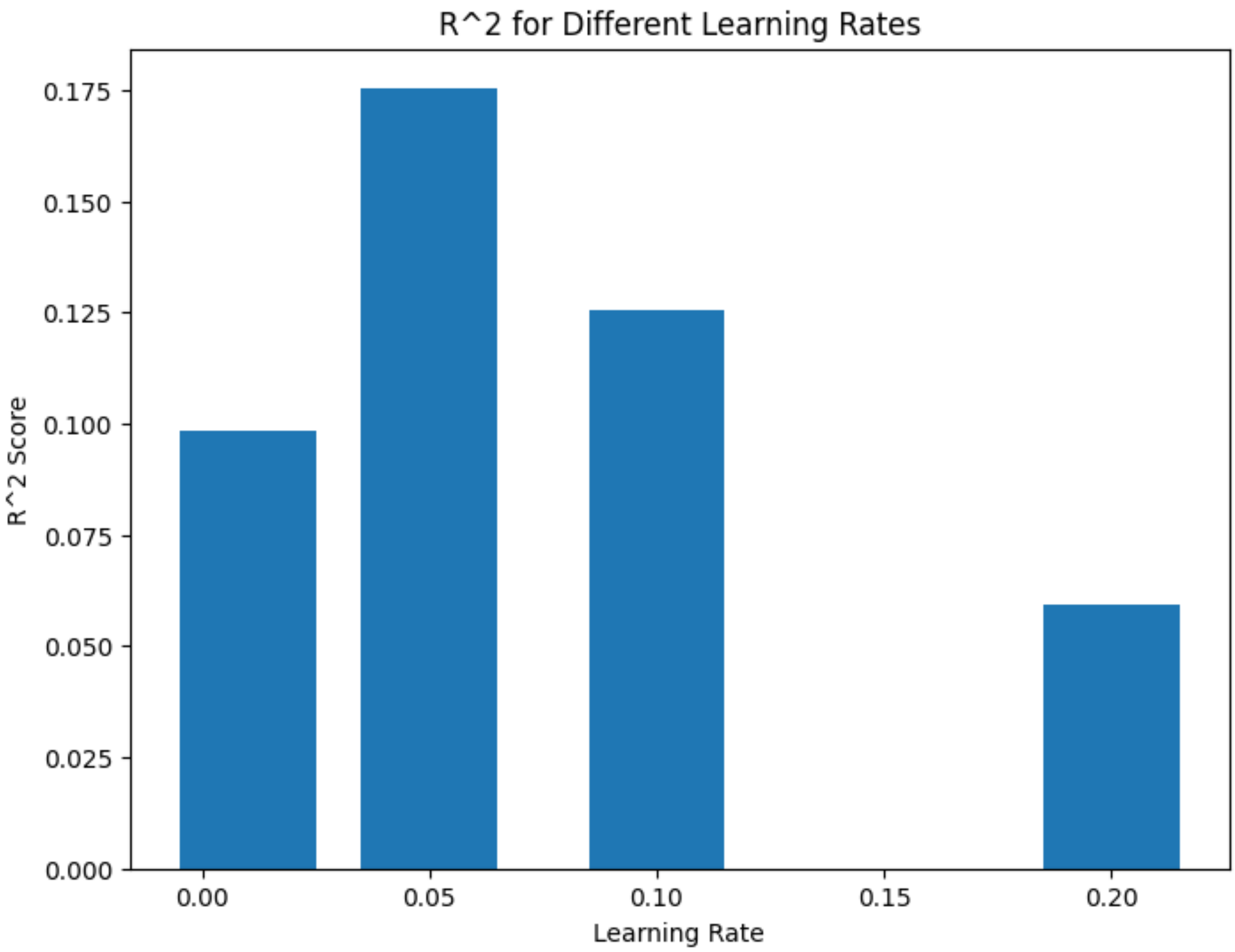


In [3]:
```
# Define learning rates to experiment with
```

```
learning_rates = [0.01, 0.05, 0.1, 0.2]

# Initialize a list to store results
results = []

for lr in learning_rates:
    # Initialize and train the Gradient Boosting Regressor with current l
    model = GradientBoostingRegressor(learning_rate=lr, random_state=42)
    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)
    results.append({'Learning Rate': lr, 'Mean Squared Error': mse, 'R^2'

    # Print results for this learning rate
    print(f"Learning Rate: {lr}")
    print(f"Mean Squared Error (MSE): {mse:.2f}")
    print(f"R-squared (R^2): {r2:.2f}")
    print('-' * 30)

# Convert results to a DataFrame for better visualization
results_df = pd.DataFrame(results)

# Display the table
print(results_df)

# Visualize the table as a bar plot for MSE
plt.figure(figsize=(8, 6))
plt.bar(results_df['Learning Rate'], results_df['Mean Squared Error'], wi
plt.xlabel('Learning Rate')
plt.ylabel('Mean Squared Error')
plt.title('MSE for Different Learning Rates')
plt.show()

# Visualize the table as a bar plot for R^2
plt.figure(figsize=(8, 6))
plt.bar(results_df['Learning Rate'], results_df['R^2'], width=0.03)
plt.xlabel('Learning Rate')
plt.ylabel('R^2 Score')
plt.title('R^2 for Different Learning Rates')
plt.show()
```

```
Learning Rate: 0.01
Mean Squared Error (MSE): 18.48
R-squared (R^2): 0.10
------------------------------
Learning Rate: 0.05
Mean Squared Error (MSE): 16.91
R-squared (R^2): 0.18
------------------------------
Learning Rate: 0.1
Mean Squared Error (MSE): 17.93
R-squared (R^2): 0.13
------------------------------
Learning Rate: 0.2
Mean Squared Error (MSE): 19.29
R-squared (R^2): 0.06
------------------------------
   Learning Rate  Mean Squared Error       R^2
0           0.01           18.484882  0.098520
1           0.05           16.906134  0.175514
2           0.10           17.928775  0.125641
3           0.20           19.290677  0.059223
```




R^2 for Different Learning Rates
R^2 Score
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0.00
0.05
0.10
0.15
0.20
Learning Rate

In [ ]: