Importing Required Libraries and Data set

In [52]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score, accurac
```

In [53]:
```
df=pd.read_csv('C:/Users/niran/Desktop/Admission_Predict.csv')
```

Data Analysis

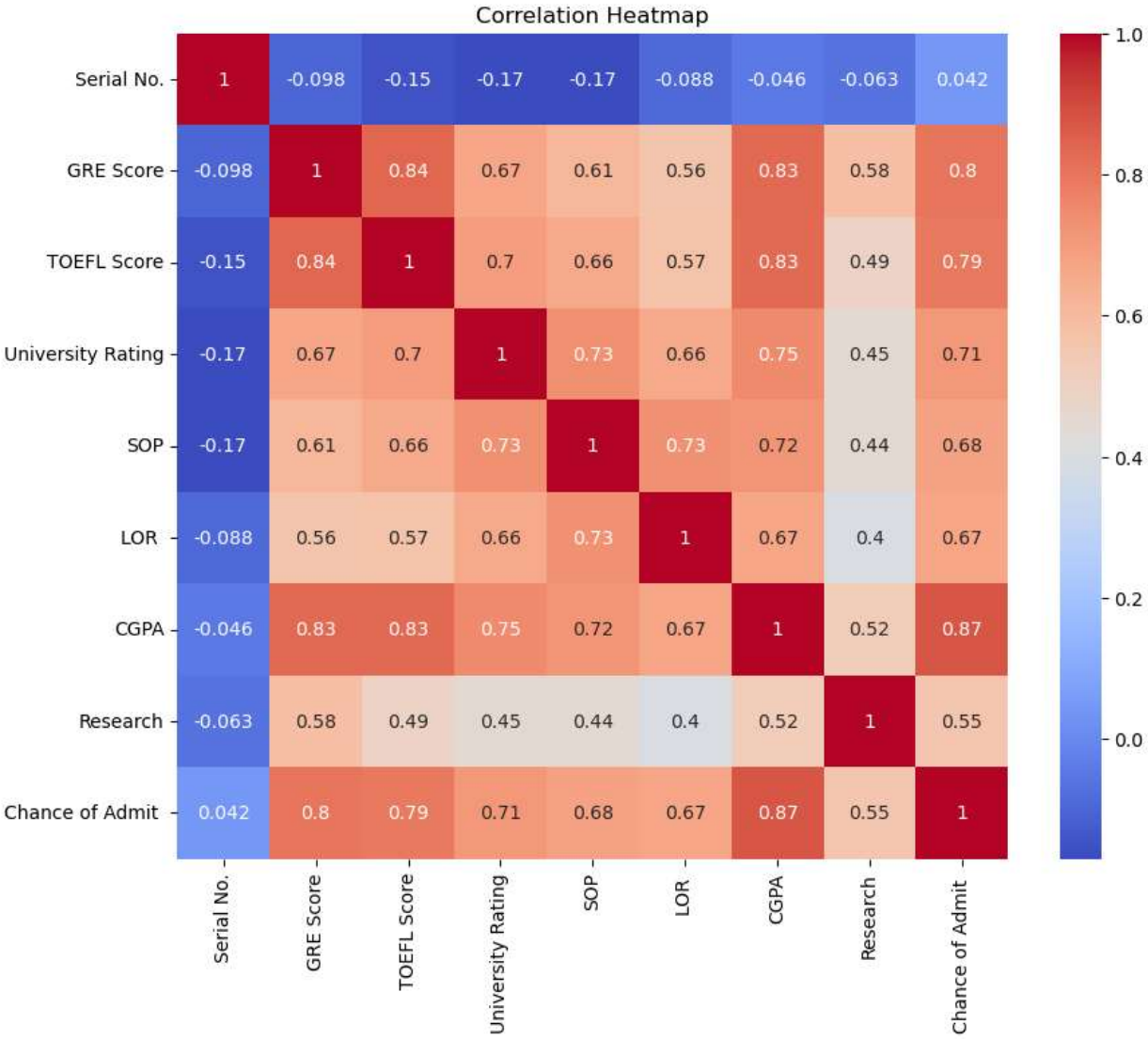
In [54]:
```
def explore_data(df):
    pd.set_option('display.max_rows',100)
    pd.set_option('display.max_columns',100)

    #Get the basic information about the dataframe
    print("Data Shape:")
    print(df.shape)

    print("\nData Columns:")
    print(df.columns)

    print("\nData Info:\n")
    print(df.info())

    #Check for missing values
    print("\nMissing values:\n")
    print(df.isnull().sum())

    #Check for duplicate rows
    print("\nDuplicate rows\n:")
    print(df.duplicated().sum())

    #Explore unique values in all columns
    print("\nUnique values in all columns\n:")
    print(df.nunique())
```

In [55]:
```
explore_data(df)
```

```
Data Shape:
(400, 9)

Data Columns:
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR ', 'CGPA', 'Research', 'Chance of Admit '],
      dtype='object')

Data Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         400 non-null    int64  
 1   GRE Score          400 non-null    int64  
 2   TOEFL Score        400 non-null    int64  
 3   University Rating  400 non-null    int64  
 4   SOP                400 non-null    float64
 5   LOR                400 non-null    float64
 6   CGPA               400 non-null    float64
 7   Research           400 non-null    int64  
 8   Chance of Admit    400 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
None

Missing values:

Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64

Duplicate rows
:
0

Unique values in all columns
:
Serial No.           400
GRE Score             49
TOEFL Score           29
University Rating      5
SOP                    9
LOR                    9
CGPA                 168
Research               2
Chance of Admit       60
dtype: int64
```

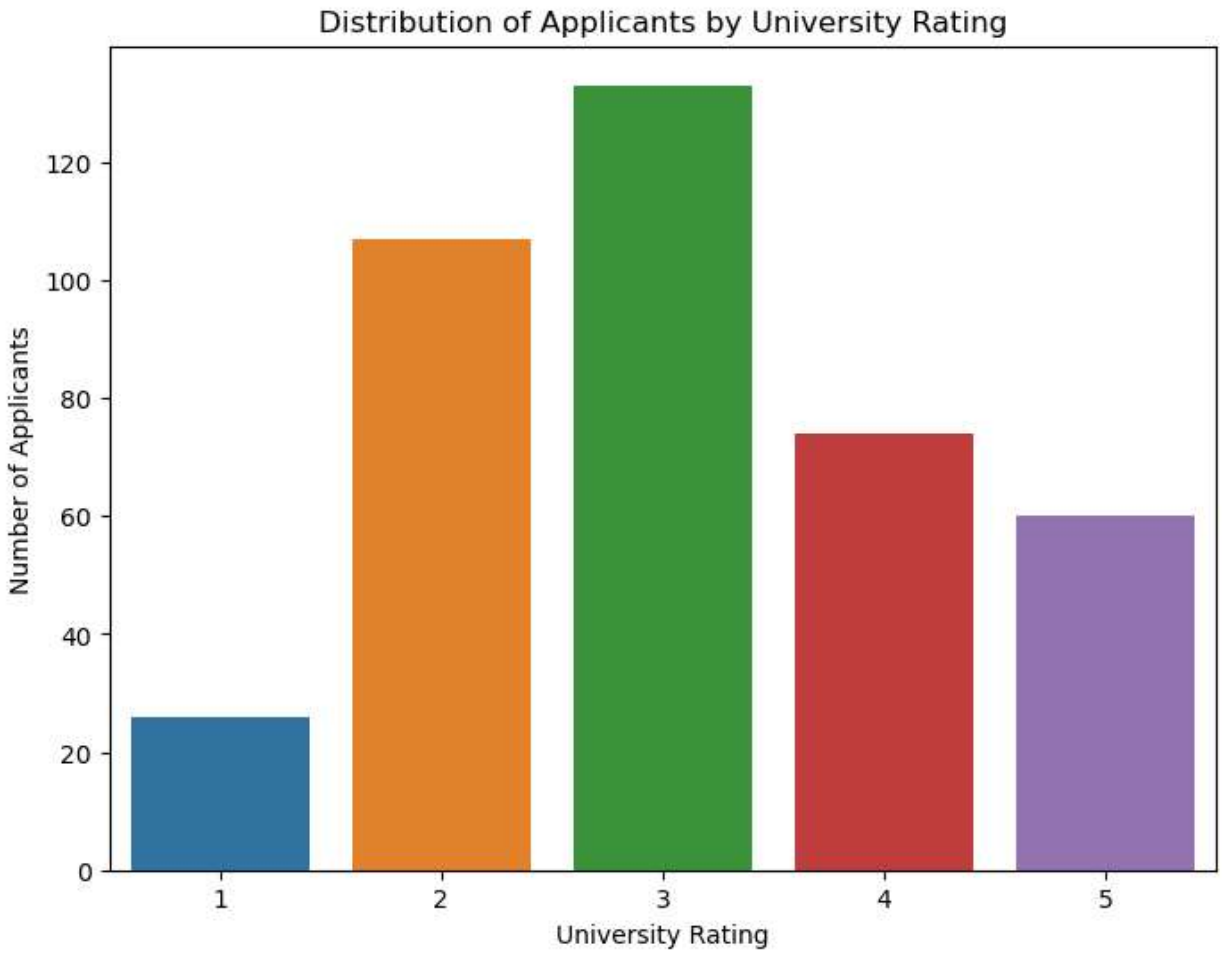
Data Visualization

In [46]:
```
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



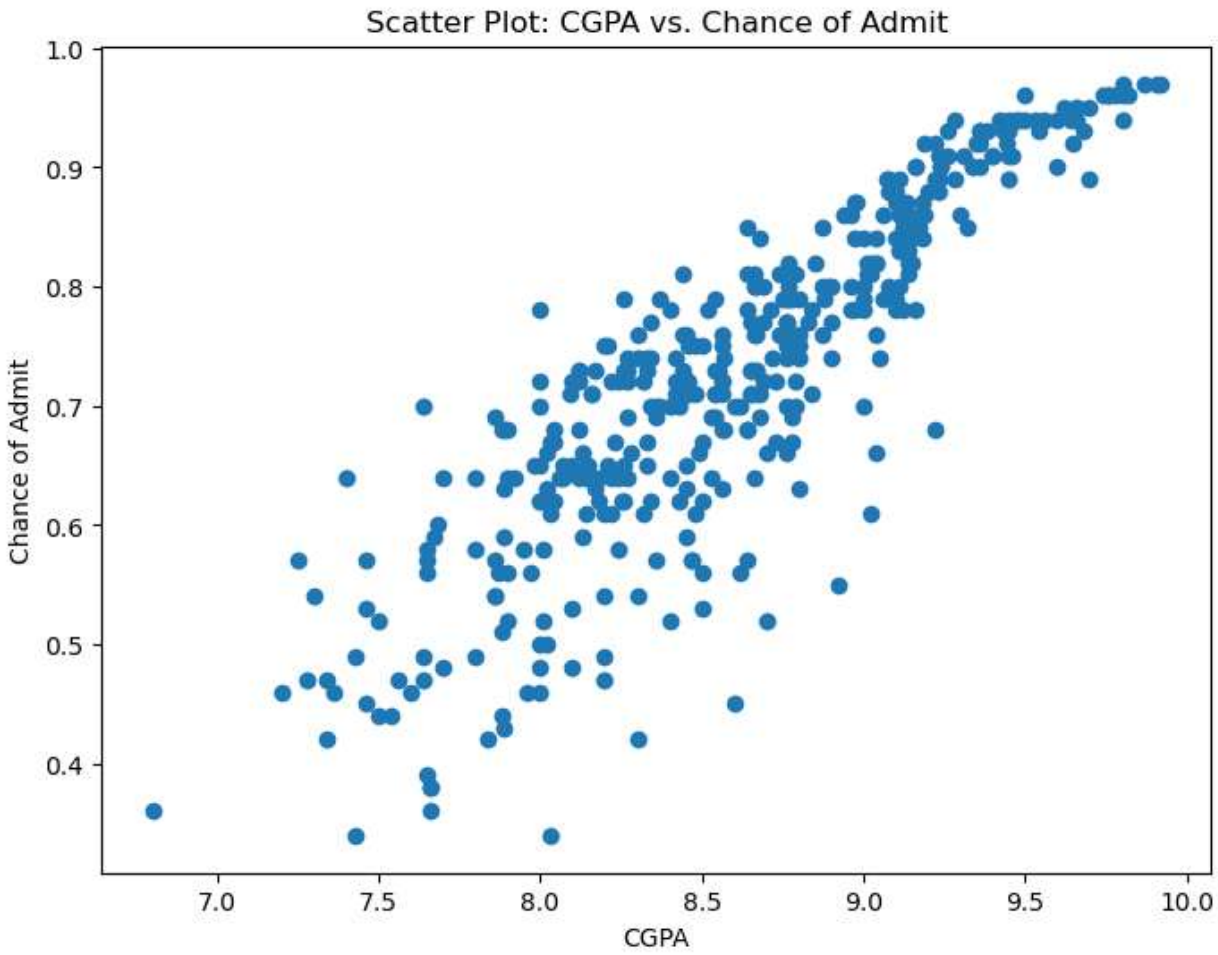
GRE Score, TOEFL Score, CGPA and Chance of Admit are highly correlated with each other

In [47]:
```
plt.figure(figsize=(8, 6))
sns.countplot(x="University Rating", data=df)
plt.title("Distribution of Applicants by University Rating")
plt.xlabel("University Rating")
plt.ylabel("Number of Applicants")
plt.show()
```



The Imbalance Is Not Significant

In [64]:
```
plt.figure(figsize=(8, 6))
plt.scatter(df["CGPA"], df["Chance of Admit "])
plt.title("Scatter Plot: CGPA vs. Chance of Admit")
plt.xlabel("CGPA")
plt.ylabel("Chance of Admit")
plt.show()
```



CGPA and Chance of Admit have a good linear relationship!

Data Preprocessing and Model Building

In [65]:
```
X=df.drop(['Chance of Admit '],axis=1)
y=df['Chance of Admit ']
```

In [66]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_stat
```

In [68]:
```
models = [
    RandomForestRegressor(),
    LinearRegression(),
    DecisionTreeRegressor(),
    GradientBoostingRegressor()
]
```

In [69]:
```
for regressor in models:
    print("\n-------------------------")
    print(f'Regressor: {regressor.__class__.__name__}')

    regressor.fit(X_train, y_train)

    pred = regressor.predict(X_test)
    mse, mae ,r2= mean_squared_error(pred, y_test), mean_absolute_error(pred, y_test),
    print(f'Mean Squared Error: {mse}\tMean Absolute Error: {mae} \tR2 score: {r2}\n\r
```

```
------------------------
Regressor: RandomForestRegressor
Mean Squared Error: 5.466625000000044e-06       Mean Absolute Error: 0.00094375000000
02555   R2 score: 0.9996990984064077



------------------------
Regressor: LinearRegression
Mean Squared Error: 6.280072362657899e-31       Mean Absolute Error: 6.45317133063372
2e-16   R2 score: 1.0



------------------------
Regressor: DecisionTreeRegressor
Mean Squared Error: 2.7499999999999998e-05      Mean Absolute Error: 0.00125000000000
00323   R2 score: 0.998486306665669



------------------------
Regressor: GradientBoostingRegressor
Mean Squared Error: 3.741332561218278e-06       Mean Absolute Error: 0.00038609520988
976755  R2 score: 0.9997940643578388
```

Model Summary

- Considering the provided metrics and the potential concerns, both the Random Forest Regressor and the Gradient Boosting Regressor appear to be strong candidates for the best model. They consistently achieved low MSE and MAE values with high R2 scores, indicating excellent predictiveperformance. *
- Between the two, Gradient Boosting Regressor might be slightly preferable due to its ensemble nature, which generally reduces overfitting. *

In [ ]: