In [1]:
```
import pandas as pd
import seaborn as sns

df = pd.read_csv('Synthetic_app_data.csv')
df
```

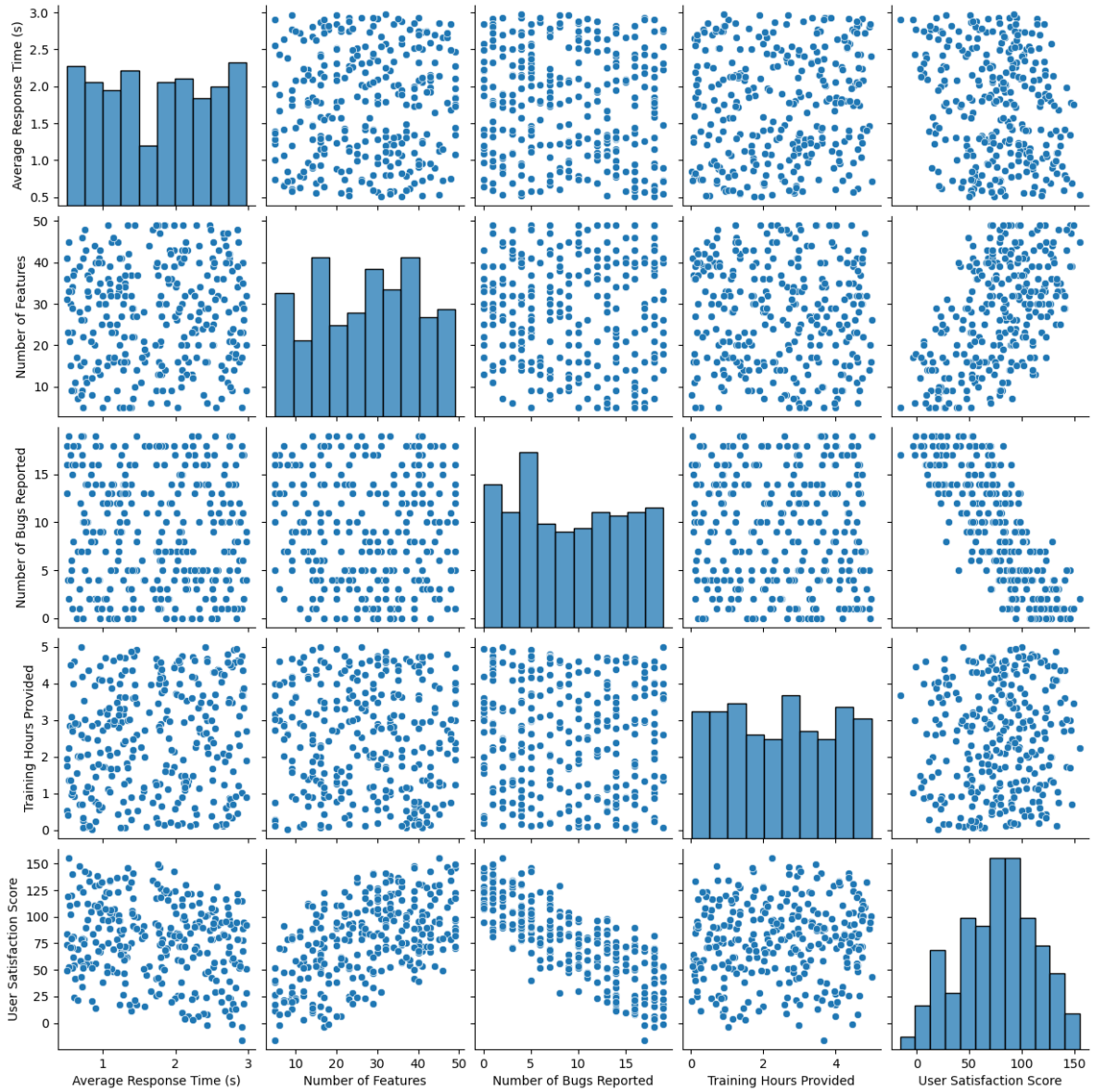
Out[1]:

	Average Response Time (s)	Number of Features	Number of Bugs Reported	Training Hours Provided	User Satisfaction Score
0	1.436350	49	16	3.970126	89.980978
1	2.876786	36	12	3.100364	56.732146
2	2.329985	34	0	2.667305	121.502856
3	1.996646	39	1	4.469463	124.535638
4	0.890047	44	8	3.942986	129.077397
...	...	...	...	...	...
295	1.805608	15	6	4.373508	75.282835
296	2.424984	25	18	2.644686	24.424736
297	1.039553	30	7	4.695338	107.672922
298	2.057226	29	9	3.993916	82.359435
299	0.713369	26	19	4.989671	43.682683

300 rows × 5 columns

In [2]:
```
sns.pairplot(df)
```

Out[2]:
```
<seaborn.axisgrid.PairGrid at 0x196bc0e7d90>
```



Obesrvations

User Satisfaction Score with Average Response Times: When we compare User Satisfaction Score with Average Response Times we do not find linearity between them as all the points are scattered.

User Satisfaction Score with Number of Features: In this case we observer that there is linearity between User Satisfaction and Number of Features as all the data that is plotted in the graph somewhat creates a linear fashion.

User Satisfaction Score with Number of Bugs Reported: In this case it is observer that there is a high linearity between the 2 attributes considered here as all the datapoints plotted here are plotted in a linear format.

User Satisfaction Score with Training Hours Provided: In the above case we obeserve that there is no linearity between the considered attributes as all the points are scattered across the

In [3]:
```
ind_variable = df[['Average Response Time (s)', 'Number of Features', 'Number
dep_variable = df['User Satisfaction Score'].values
```

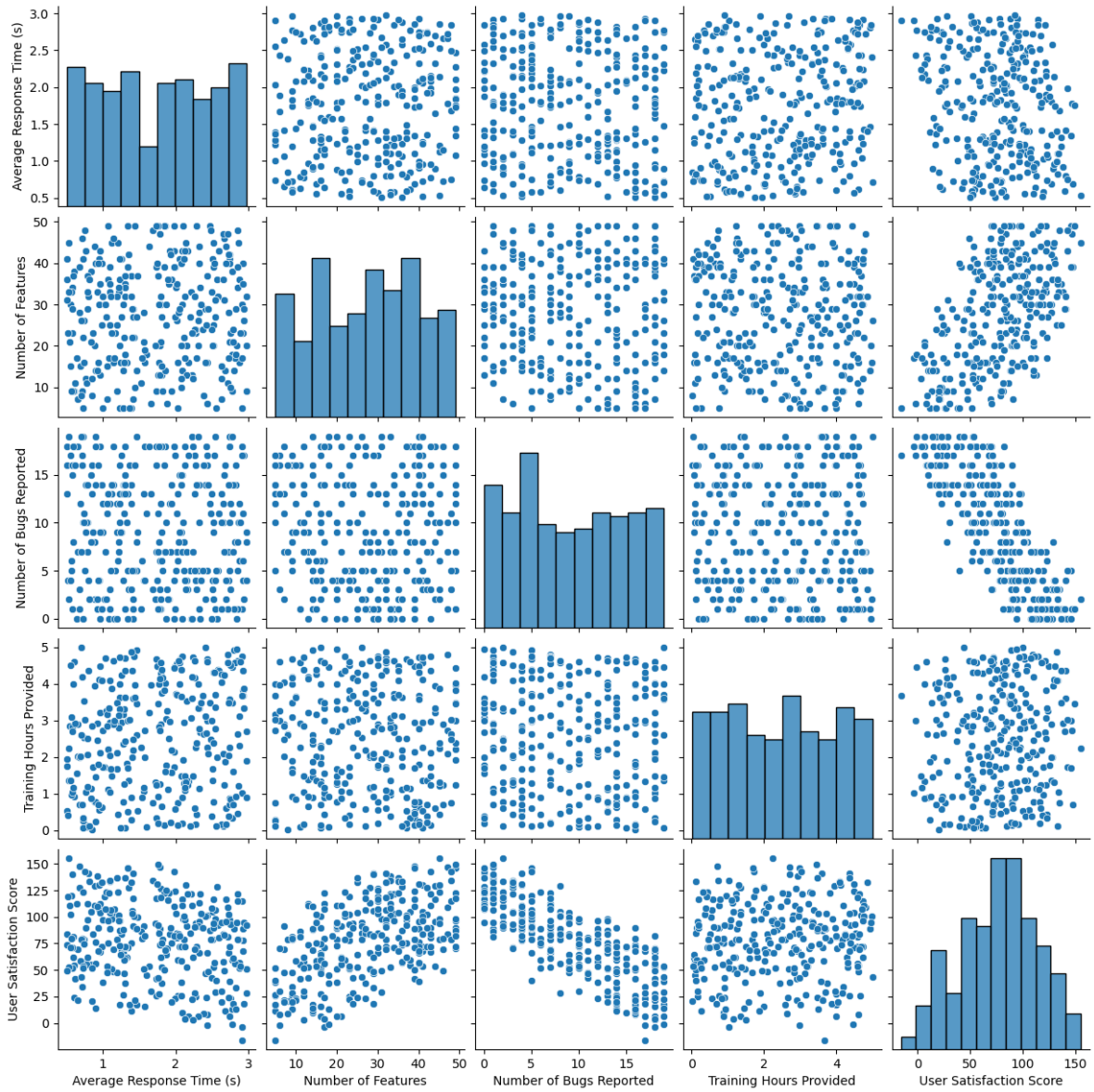
In [4]:
```
ind_variable
```

Out[4]:

	Average Response Time (s)	Number of Features	Number of Bugs Reported	Training Hours Provided
0	1.436350	49	16	3.970126
1	2.876786	36	12	3.100364
2	2.329985	34	0	2.667305
3	1.996646	39	1	4.469463
4	0.890047	44	8	3.942986
...	...	...	...	...
295	1.805608	15	6	4.373508
296	2.424984	25	18	2.644686
297	1.039553	30	7	4.695338
298	2.057226	29	9	3.993916
299	0.713369	26	19	4.989671

300 rows × 4 columns

In [22]:
```
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt

X_train, X_test, y_train, y_test = train_test_split(ind_variable, dep_variable

model = LinearRegression()
model.fit(X_train, y_train)

y_predictions = model.predict(X_test)

sns.pairplot(df)
plt.show()

#Print y intercept
print(f"Intercept (Bias) : {model.intercept_}\n")

#Coefficients
print('Coefficients:')
coef_arr = model.coef_

print(f"Average Response Time (s) : {coef_arr[0]}")
print(f"Number of Features : {coef_arr[1]}")
print(f"Number of Bugs Reported : {coef_arr[2]}")
print(f"Training Hours Provided : {coef_arr[3]}")

print('\nEvaluation Matrix on Test Set:')


#Calculate Mean Absolute Error
mae = metrics.mean_absolute_error(y_test, y_predictions)
print(f"Mean Absolute Error (MAE) : {mae}")
#Calculate MSE

mse = metrics.mean_squared_error(y_test, y_predictions)
print(f"Mean Square Error (MSE) : {mse}")

root_mse = np.sqrt(metrics.mean_squared_error(y_test, y_predictions))
print(f"Root Mean Square Error (RMSE) : {root_mse}")

r2squared = metrics.r2_score(y_test, y_predictions)
print(f"RSquared (R2) : {r2squared}")
```




```
Intercept (Bias) : 98.9806165078723

Coefficients:
Average Response Time (s) : -14.99444607358292
Number of Features : 1.492636196962363
Number of Bugs Reported : -4.889397201108882
Training Hours Provided : 3.0112360497844555

Evaluation Matrix on Test Set:
Mean Absolute Error (MAE) : 4.244209304433561
Mean Square Error (MSE) : 28.220726519405183
Root Mean Square Error (RMSE) : 5.312318375192246
RSquared (R2) : 0.9830763925622145
```

In [ ]: