Importing Relevant Libraries

In [63]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import gensim
from gensim import corpora
from gensim.models import LdaModel
import pprint
from textblob import TextBlob
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

In [64]:
```
# !pip install openpyxl
# !pip install textblob
# !pip install gensim
# nltk.download('punkt')
# nltk.download('stopwords')
# nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\jagme\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\jagme\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\jagme\AppData\Roaming\nltk_data...
```

Out[64]:
```
True
```

Data Preprocessing

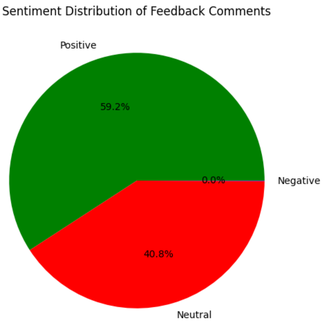
In [65]:
```
# Load the Excel file
excel_file = "AI_Engineer_Dataset_Task_1.xlsx"
df = pd.read_excel(excel_file)

# Assuming the text data is in a column named "text_column"
text_data = df["ParticipantResponse"]

# Function to preprocess text using NLTK
def preprocess_text_nltk(text):
    if isinstance(text, str):
        # Convert text to lowercase
        text = text.lower()
        # Tokenize text using NLTK
        tokens = word_tokenize(text)
        # Remove punctuation, special characters, and digits
        tokens = [word for word in tokens if word.isalpha()]
        # Remove stopwords using NLTK
        stop_words = set(stopwords.words('english'))
        tokens = [word for word in tokens if word not in stop_words]
        # Lemmatize words using NLTK
        lemmatizer = WordNetLemmatizer()
        tokens = [lemmatizer.lemmatize(word) for word in tokens]
        # Reconstruct the text
        text = ' '.join(tokens)
    else:
        # If it's not a string, convert it to an empty string
        text = ""
    return text

# Apply the preprocessing function to the text data
text_data_preprocessed_nltk = text_data.apply(preprocess_text_nltk)

# Display the preprocessed text data
print(text_data_preprocessed_nltk)
```

```
0                 disagree
1         strongly disagree
2         strongly disagree
3         strongly disagree
4         strongly disagree
                ...
180964
180965                agree
180966       strongly agree
180967       strongly agree
180968       strongly agree
Name: ParticipantResponse, Length: 180969, dtype: object
```

Sentiment Analysis

In [66]:
```
# function to assign sentiment
def get_sentiment_polarity(text):
    analysis = TextBlob(text)
    if analysis.sentiment.polarity > 0:
        return "Positive"
    elif analysis.sentiment.polarity < 0:
        return "Negative"
    else:
        return "Neutral"

# Apply sentiment analysis to each comment
text_data_preprocessed_nltk = text_data_preprocessed_nltk.astype(str)  # Ensure all values are strings
text_data['Sentiment'] = text_data_preprocessed_nltk.apply(get_sentiment_polarity)

# Calculate the distribution of sentiment
sentiment_distribution = text_data['Sentiment'].value_counts()

# Create a pie chart to visualize sentiment distribution
plt.figure(figsize=(8, 6))
sentiment_distribution.plot(kind='pie', autopct='%1.1f%%', colors=['green', 'red', 'blue'])
plt.title('Sentiment Distribution of Feedback Comments')
plt.ylabel('')
plt.show()

# Display summary statistics
print("Sentiment Distribution Summary:")
print(sentiment_distribution)
```

```
C:\Users\jagme\AppData\Local\Temp\ipykernel_1576\2579182003.py:41: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  text_data['Sentiment'] = text_data_preprocessed_nltk.apply(get_sentiment_polarity)
```




```
Sentiment Distribution Summary:
ParticipantResponse
Positive    107065
Neutral      73819
Negative        85
Name: count, dtype: int64
```

Topic Modeling

In [67]:
```
# Tokenize the preprocessed text data
tokenized_data = text_data_preprocessed_nltk.apply(lambda x: x.split())

# Create a dictionary from the tokenized data
dictionary = corpora.Dictionary(tokenized_data)

# Create a document-term matrix (corpus)
corpus = [dictionary.doc2bow(text) for text in tokenized_data]

# Define the number of topics for LDA
num_topics = 5
```

In [68]:
```
# Create the LDA model
lda_model = LdaModel(corpus, num_topics=num_topics, id2word=dictionary, passes=5, random_state=42)

# Print the topics and their representative keywords
topics = lda_model.print_topics(num_words=5)

print("Identified Topics:")
for topic in topics:
    print(topic)
```

```
Identified Topics:
(0, '0.062*"course" + 0.040*"good" + 0.022*"dr" + 0.019*"thank" + 0.019*"best"')
(1, '0.573*"agree" + 0.422*"strongly" + 0.000*"sss" + 0.000*"زیرمجموعه" + 0.000*"."')
(2, '0.031*"0.024 + "ف"*0.024 + "ر"*0.027 + "ا"*"great" + 0.023*"س"')
(3, '0.567*"disagree" + 0.408*"strongly" + 0.000*"sss" + 0.000*"زیرمجموعه" + 0.000*"."')
(4, '0.936*"neutral" + 0.004*"none" + 0.000*"sss" + 0.000*"زیرمجموعه" + 0.000*"."')
```

Insights and Recommendations

In [69]:
```
print("\nTopic Modeling Insights:")
topics = lda_model.print_topics(num_words=5)
for i, topic in enumerate(topics):
    print(f"Topic {i+1}: {topic}")
```

```
Topic Modeling Insights:
Topic 1: (0, '0.062*"course" + 0.040*"good" + 0.022*"dr" + 0.019*"thank" + 0.019*"best"')
Topic 2: (1, '0.573*"agree" + 0.422*"strongly" + 0.000*"sss" + 0.000*"زیرمجموعه" + 0.000*"."')
Topic 3: (2, '0.031*"0.024 + "ف"*0.024 + "ر"*0.027 + "ا"*"great" + 0.023*"س"')
Topic 4: (3, '0.567*"disagree" + 0.408*"strongly" + 0.000*"sss" + 0.000*"زیرمجموعه" + 0.000*"."')
Topic 5: (4, '0.936*"neutral" + 0.004*"none" + 0.000*"sss" + 0.000*"زیرمجموعه" + 0.000*"."')
```

In [70]:
```
positive_count = sentiment_distribution.get("Positive", 0)
negative_count = sentiment_distribution.get("Negative", 0)
neutral_count = sentiment_distribution.get("Neutral", 0)
```

In [71]:
```
print("\nRecommendations:")
if positive_count > negative_count:
    print("Based on the sentiment analysis, high number of positive comments are identified , indicating areas of strength.")
else:
    print("Based on the sentiment analysis, high number of Negative comments are identified, indicating potential areas for improvement.")
```

```
Recommendations:
Based on the sentiment analysis, there are more positive comments, indicating areas of strength.
```

In [ ]:
```

```