Class Exercise:

Files v3

You are working as a data engineer for a social media platform. Your task is to analyze user engagement data for posts. You have been provided with a file named "user\_engagement.txt" that contains the user engagement data in the following format:

**Post ID | Likes | Comments | Shares**

Your goal is to create a Python class called **"EngagementAnalyzer"** that performs the following operations:

1. Read the user engagement data from the file using a file iterator.
2. Calculate the total number of likes, comments, and shares across all posts.
3. Find the post with the highest engagement (sum of likes, comments, and shares).
4. Determine the average engagement per post.
5. Write the analysis results to a new file named **"engagement\_analysis.txt"**.
6. Provide a function to search for a specific post by its ID and retrieve its engagement metrics.
7. Implement a function to calculate the engagement rate, defined as the average engagement per post divided by the number of followers.
8. Add a function to calculate the engagement ratio, defined as the number of comments divided by the sum of likes and shares for each post.
9. Write the engagement metrics of all posts to a separate file named **"post\_engagement\_metrics.txt"**.

class EngagementAnalyzer:

    def \_\_init\_\_(self, engagement\_file\_path):

# Your code here

    def load\_engagement\_data(self):

# Your code here

    def calculate\_total\_engagement(self):

# Your code here

    def find\_highest\_engagement\_post(self):

# Your code here

    def calculate\_average\_engagement(self):

# Your code here

    def search\_post\_by\_id(self, post\_id):

# Your code here

    def calculate\_engagement\_rate(self, followers):

# Your code here

    def calculate\_engagement\_ratio(self):

# Your code here

    def write\_analysis\_results(self):

# Your code here

    def write\_post\_engagement\_metrics(self):

# Your code here

# Usage example:

analyzer = EngagementAnalyzer('user\_engagement.txt')

analyzer.write\_analysis\_results()

analyzer.write\_post\_engagement\_metrics()

# Example usage of additional functions

post\_id = '123456'

post\_metrics = analyzer.search\_post\_by\_id(post\_id)

if post\_metrics:

    print(f"Engagement metrics for post ID {post\_id}: {post\_metrics}")

else:

    print(f"Post with ID {post\_id} not found.")

followers\_count = 1000

engagement\_rate = analyzer.calculate\_engagement\_rate(followers\_count)

print(f"Engagement rate: {engagement\_rate:.2f}")

engagement\_ratios = analyzer.calculate\_engagement\_ratio()

print(f"Engagement ratios: {engagement\_ratios}")

Expected output:

**engagement\_analysis.txt:**

Total Likes: 236

Total Comments: 72

Total Shares: 54

Highest Engagement Post ID: ABC123

Average Engagement per Post: 122.67

**post\_engagement\_metrics.txt:**

Post ID: XYZ789

Likes: 100

Comments: 20

Shares: 10

Engagement Ratio: 0.17

Post ID: ABC123

Likes: 120

Comments: 40

Shares: 44

Engagement Ratio: 0.41

**Output of additional functions:**

Engagement metrics for post ID 123456: {'post\_id': '123456', 'likes': 80, 'comments': 30, 'shares': 20}

Engagement rate: 0.12

Engagement ratios: [0.13, 0.23]