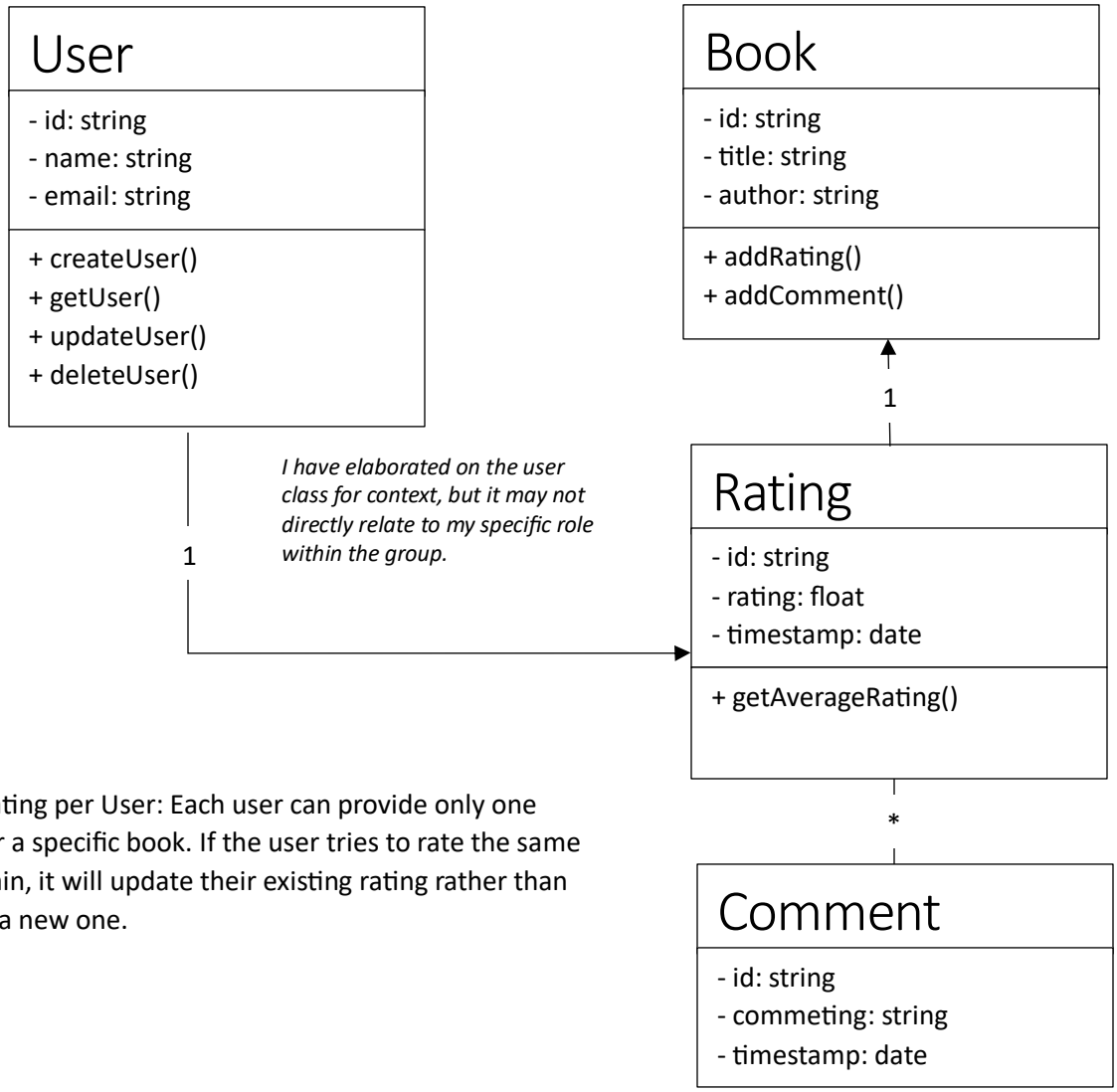
UML Diagrams

Anthony Carvalho (Group 4)

Book Rating and Commenting

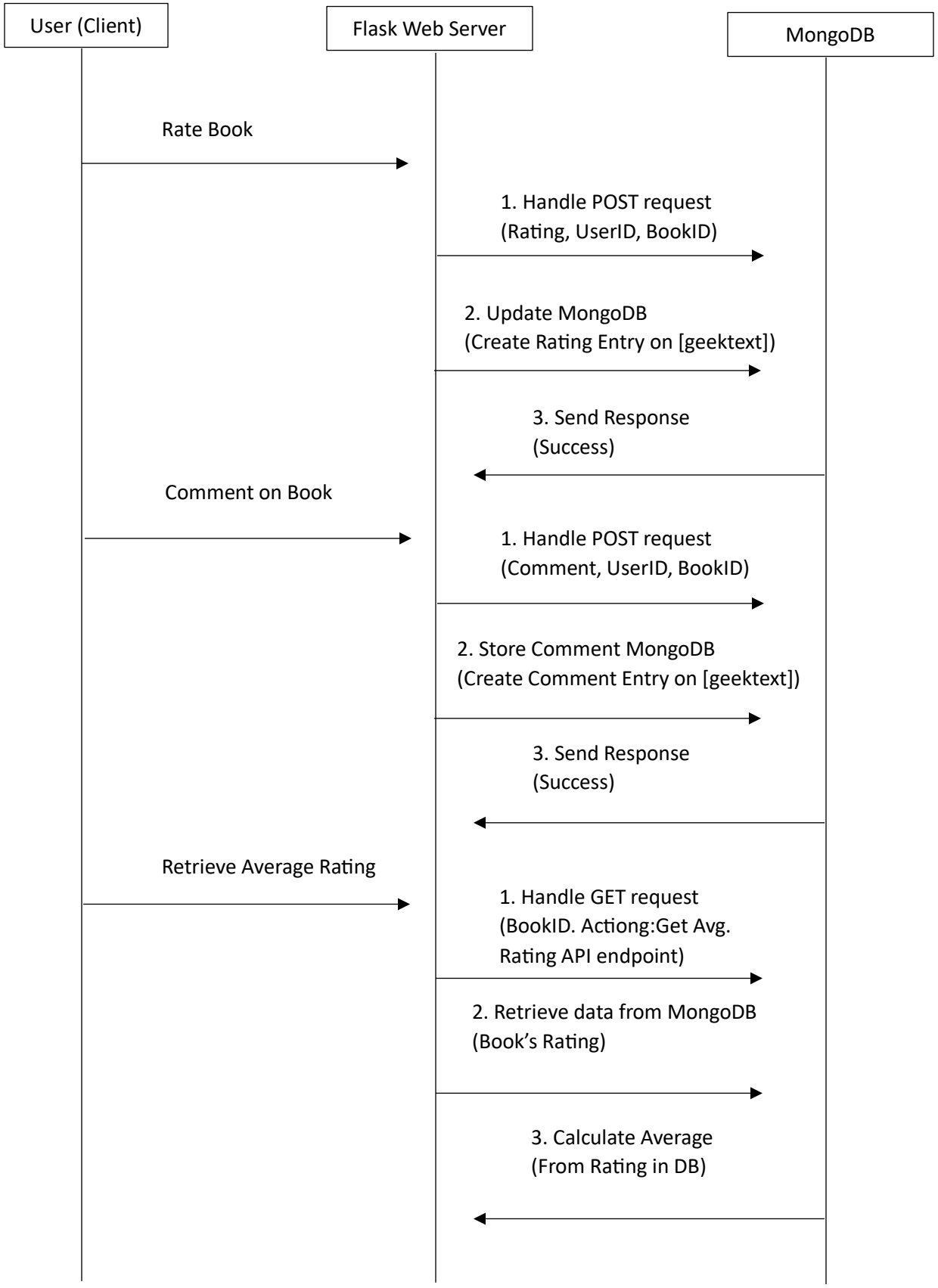
Class Diagram with Methods Attributes, Associations and Multiplicity



1 One Rating per User: Each user can provide only one rating for a specific book. If the user tries to rate the same book again, it will update their existing rating rather than creating a new one.

* Multiple Comments per Book: Multiple users can provide multiple comments for the same book. There are no limitations on the number of comments that different users can add to a particular book.

Commenting and Rating, Interaction Flow: Client-Flask API-Database Sequence Diagram



Breakdown on Interaction Flow in Book Rating and Commenting System

This document outlines the sequence of interactions within the Book Rating and Commenting System. It describes how users can rate and comment on books, retrieve comments, and obtain average ratings through specific HTTP requests. The interactions involve various system components, including the User, Controller, Rating Service, Comment Service, and MongoDB. The diagram illustrates the flow of messages, the activation of services, and the successful responses back to the User.

1. User Rates a Book:

User sends a **POST** request to the **Controller** with rating details.
Controller activates the **Rating Service**.
Rating Service updates the rating in the **MongoDB** database.
Controller responds with success message to **User**.

2. User Comments on a Book:

User sends a **POST** request to the **Controller** with comment details.
Controller activates the **Comment Service.**
Comment Service adds the comment to the **MongoDB** database.
Controller responds with success message to **User.**

3. Retrieve Comments for a Book:

User sends a **GET** request for comments to the **Controller**.
Controller activates the **Comment Service.**
Comment Service fetches comments from **MongoDB.**
Controller sends JSON list of comments to **User**.

4. Retrieve Average Rating for a Book:

User sends a **GET** request for average rating to the **Controller**.
Controller activates the **Rating Service**.
Rating Service calculates average rating from **MongoDB** data.
Controller sends computed average rating to **User**.

Key Elements:

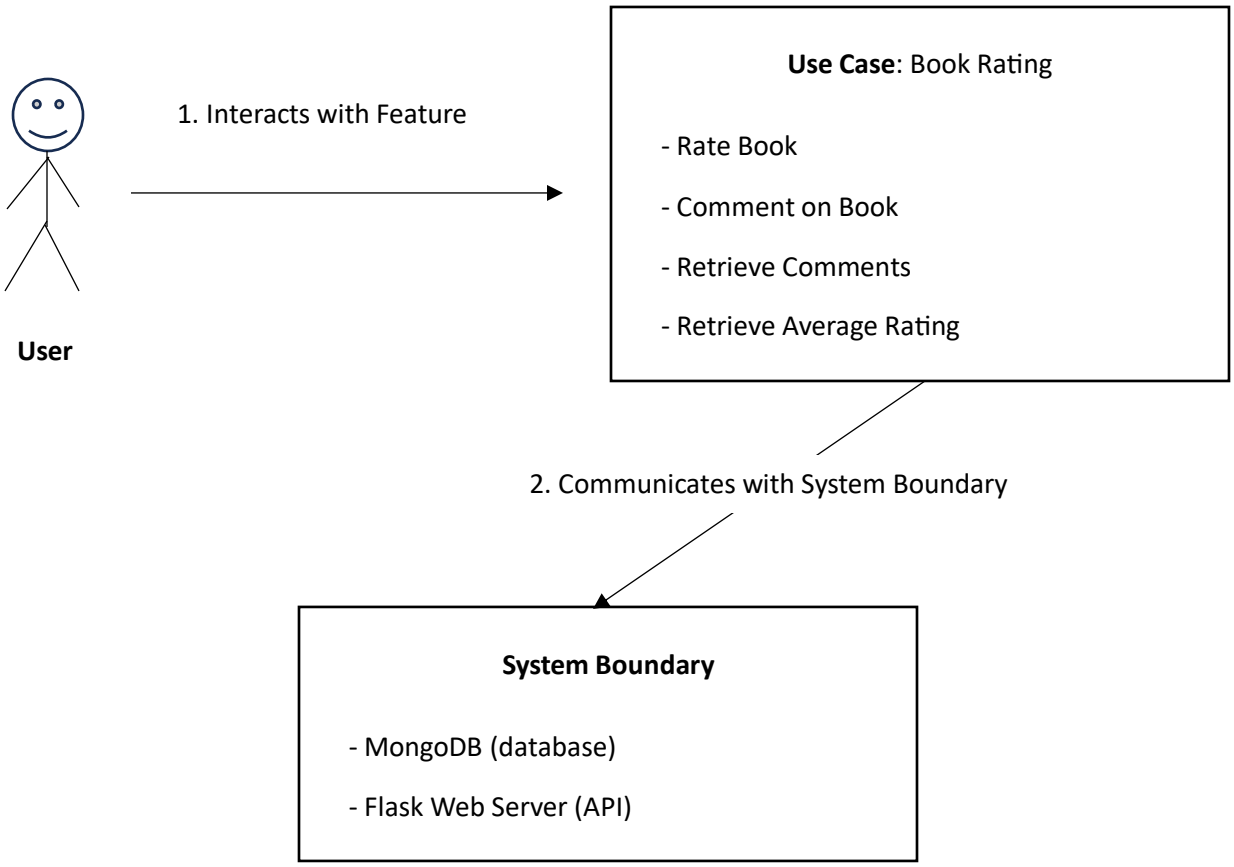
Actor: User

Activation Boxes: Controller, Rating Service, Comment Service

Lifelines: User, Controller, Rating Service, Comment Service, MongoDB

Messages: POST (for rating and commenting), GET (for retrieving comments and average rating)

Book Rating System: User Interactions and System Components Use Case Diagram



Actor: The actor in this use case is the "User," representing individuals interacting with the system.

Use Case: The primary use case is "Book Rating," which includes actions like "Rate Book," "Comment on Book," "Retrieve Comments," and "Retrieve Average Rating."

System Boundary: The system boundary includes essential components: the MongoDB database (for data storage) and the Flask Web Server (for handling API requests).