Chapter 3: Admin Backend

Ticket: User Report

Back to the Question

This is a possible implementation for this ticket:

```
public List<Critic> mostActiveCommenters(){
    List<Critic> mostActive = new ArrayList<>();

    /**
     * In this method we can use the $sortByCount stage:
     *
https://docs.mongodb.com/manual/reference/operator/aggregation/
sortByCount/index.html
     * using the $email field expression.
     */
    Bson groupByCountStage = Aggregates.sortByCount("$email");
    // Let's sort descending on the `count` of comments
    Bson sortStage =
Aggregates.sort(Sorts.descending("count"));
    // Given that we are required the 20 top users we have to
also $limit
    // the resulting list
    Bson limitStage = Aggregates.limit(20);
    // Add the stages to a pipeline
    List<Bson> pipeline = new ArrayList<>();
    pipeline.add(groupByCountStage);
    pipeline.add(sortStage);
    pipeline.add(limitStage);
    // We cannot use the CommentDao class `commentCollection`
object
    // since this returns Comment objects.
    // We need to create a new collection instance that returns
    // Critic objects instead.
    // Given that this report is required to be accurate and
    // reliable, we want to guarantee a high level of
durability, by
    // ensuring that the majority of nodes in our Replica Set
    // acknowledged all documents for this query. Therefore we
will be
    // setting our ReadConcern to "majority"
    //
https://docs.mongodb.com/manual/reference/method/cursor.readCon
cern/
    MongoCollection<Critic> commentCriticCollection =
            this.db.getCollection("comments", Critic.class)
                    .withCodecRegistry(this.pojoCodecRegistry)
                    .withReadConcern(ReadConcern.MAJORITY);

    // And execute the aggregation command output in our
collection object.

commentCriticCollection.aggregate(pipeline).into(mostActive);
    return mostActive;
}
```

Proceed to next section