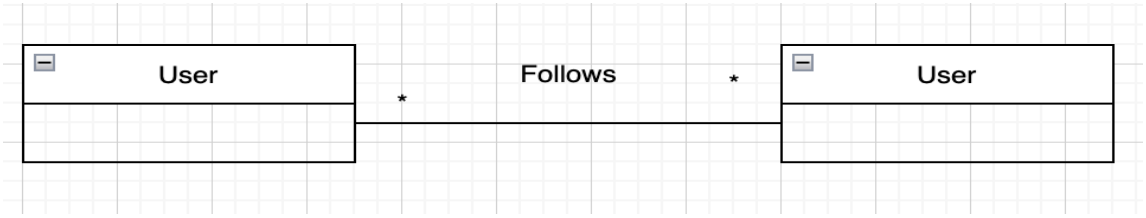
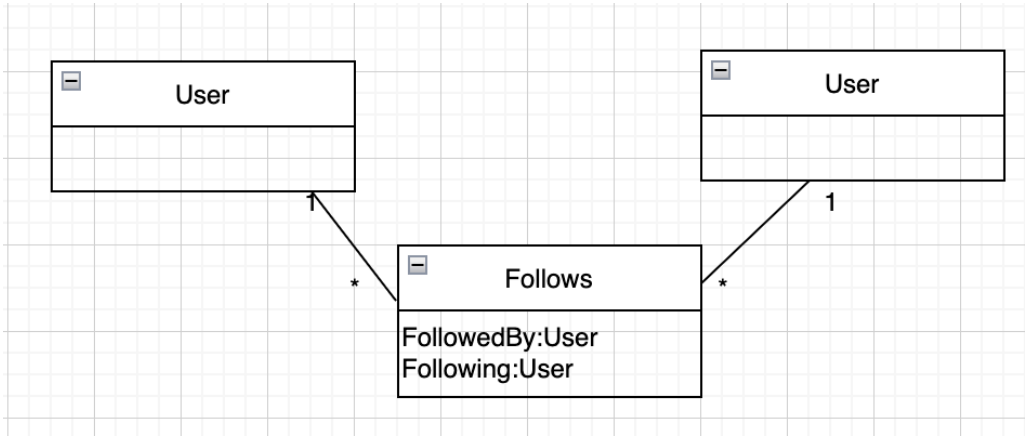
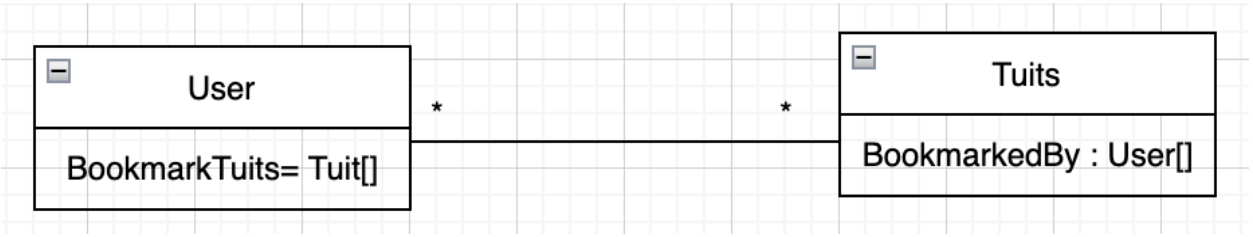
1. **Follows**
    a) This option tells us that there is a many to many relationship between users. That is, one user can follow many people and one follower can be followed by many users. But this is quite obvious and does not tell anything new.



    b) I chose this implementation because it is a mapping between the users being followed by and the user themselves following other users. This option is the most versatile, but might complicate joining in a non relational database and is more clear than option a. It clearly shows the relationship and maintains good database practices and is easier to modify later on.

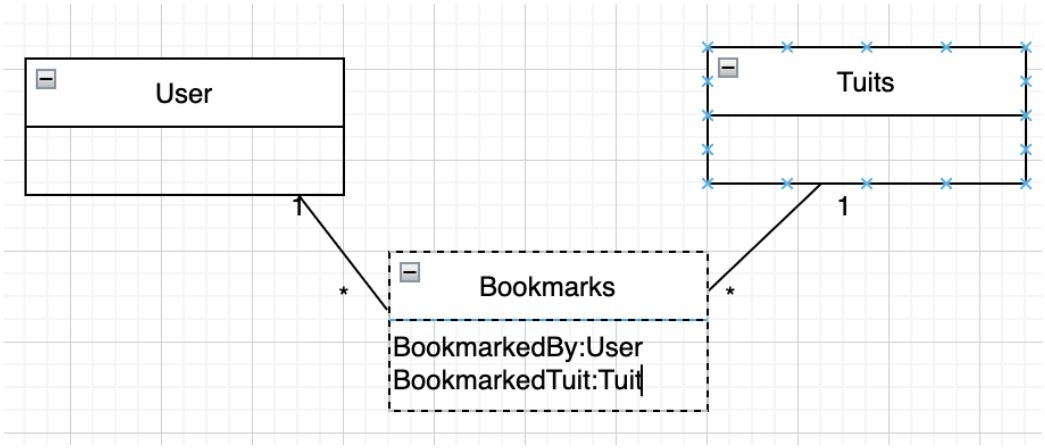


2. **Bookmarks**
    a) The first option was to save all the bookmarked tuits under users as an array and save all the users who bookmarked the tuits under tuits as another array. But this almost defeats the purpose of mongo and object oriented programming because we always need to maintain these arrays and there can be no inconsistencies.

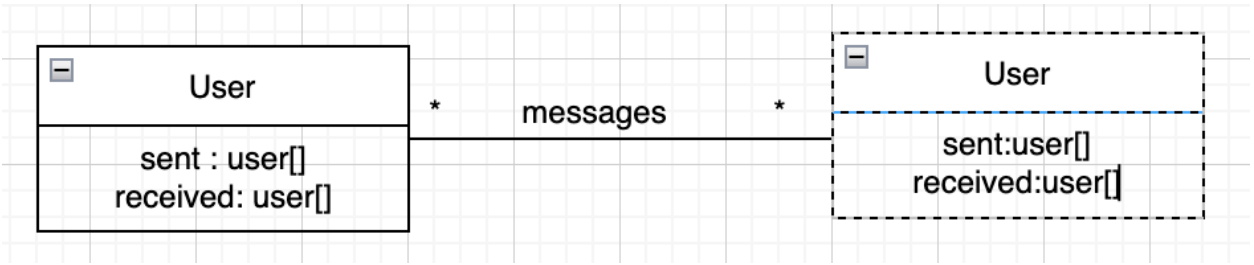


    b) The option i implemented maintains a separate bookmarks table that gets information from users and tuits. It is a mapping between user and tuit databases. It is an easier

implementation than option a and is scalable.



3. **Messages**
   a) The first option would be to save all messages sent and recieved by user 1 as separate arrays under user1 and save all messages sent and received by user 2 as arrays under user 2. But this requires a lot of memory, it is not scalable in case of broadcast messages and requires a lot of maintenance for the consistency of both arrays and need to be constantly updated.



   b) The option I chose to implement is to have a messages database as a mapping between users. This option is more versatile and does not need to be maintained as much as the

option a and is easily scaled.