Question 4

In order to add a follow feature the model would first have to be adjusted as shown:

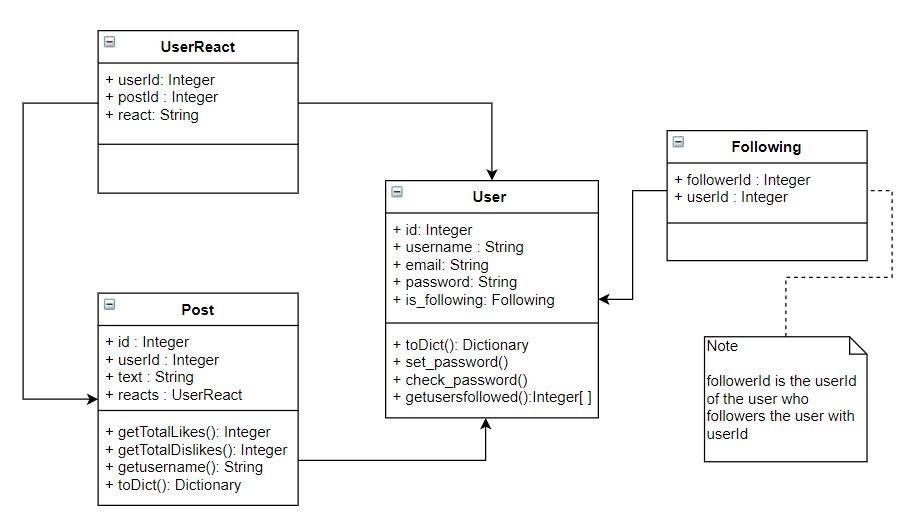


Figure : showing the adjusted model including a new Following table

Here a following table is added to keep track of which who users follow. The followerId here contains the userId of a follower and userId contains the userId of a user whose posts are to be viewed by the follower. This was each follower can follow multiple users.

The User model would be changed to accommodate the new relationship with Following. A new method would be added: getusersfollowed which returns a list of the userIds of all the users that the current user follows. It would do so by searching and extracting all the userId values associated with each matching followerId. The toDict of the user model would be changed to include the new information.

On the frontend a Follow button (or similar implementation) would be added where it can be linked to a user, next to their username in posts for example. The button would send the userId of the user to be followed and a new and a new record would be added to the Following table storing the current user as a follower of the user followed.

@app.route('/myfriendspace/follow/<userid>', methods=['GET'])  
@login\_required  
def follow(userid):  
 newfollow = Following(followerid=current\_user.id, userId=userid)  
 try:  
 db.session.add(newfollow)  
 db.session.commit()  
 return redirect(url\_for('loadhome')) # displays the updated homepage  
  
 except IntegrityError:  
 # Insert error handling for:  
 # attempting to follow the same person twice  
 # attempting to follow yourself  
 # other weird stuff  
 pass

The route that renders the home page with all the posts displayed would be changed to filter the posts according to who the current user’s follows using the new getusersfollowed method:

@app.route('/myfriendspace', methods=['GET'])  
@login\_required  
def loadhome():  
 posts = Post.query.all()  
 following = current\_user.getusersfollowed  
 posts = posts[::-1] # Lists the last post added first  
 posts = [p.toDict() for p in posts if p.userId in following] # Filtering by who the user follows  
 return render\_template("app.html", posts=posts)

Please note that the code snippets above are subject to syntax corrections and are only included to illustrate the solution idea.

Extra stuff:

|  |  |
| --- | --- |
| FollowerId (username shown for clarity) | userId (username shown for clarity) |
| 1 (Bob) | 2 (Alice) |
| 1 (Bob) | 3 (Patrick) |
| 2 (Alice) | 1 (Bob) |
| 3 (Patrick) | 1 (Bob) |
| 3 (Patrick) | 2 (Alice) |
| 1 (Bob) | 4 (Susan) |

Figure 2: I drew the table when trying to figure out the problem, decided to leave it for clarity