Question 4

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

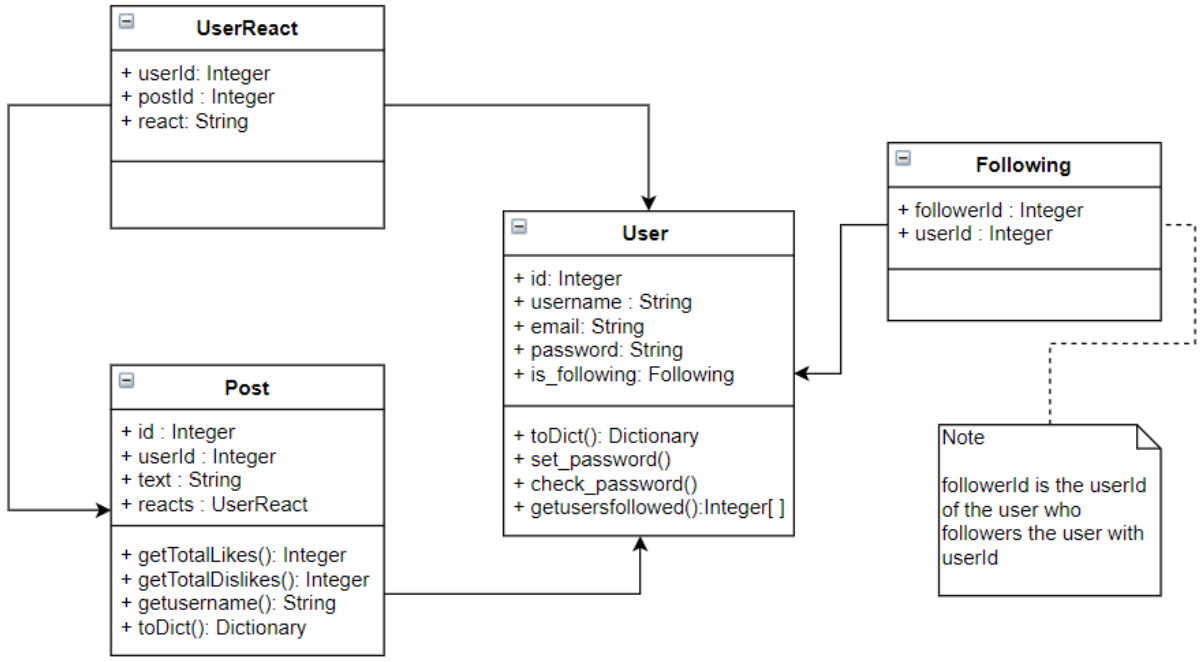

Figure 1: showing the adjusted model including a new Following table

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

Question 5

To build a Public Transport Mapper Web Application:

On the frontend I would use HTML as the base of the web pages and CSS with external libraries like Bootstrap and Materialize for styling. React a JavaScript Library would be used with JavaScript to create an intuitive user interface to improve the accessibility to users. React would allow the webpages to be updated quickly to reflect changes and make the user's experience better. These pages would also be responsive so that they are suitable for users accessing the application via mobile device.

These webpages would be dynamically served using a Flask application, the flask applications would be used for the bulkier operations like retrieving search results from the database and results from the external APIs integrated. It would also be used for CRUD (Create, Read, Update, Delete) operations on the webpages.

I would use a cloud hosted MongoDB database for persistence since it is a real-time database with high performance capabilities. A public transportation app is expected to be used by many users simultaneously and the database would have to be reliable and dependable. It would be used to store images uploaded, routes created by users and to securely stored user data generated data.

The Google Maps API and Directions API would be integrated and used along with React to create and display routes. It can also be used to search for directions and adapted to support public transport routes created. Google maps also has several additional services which can be used.

I would host the entire application including the database on Heroku preliminarily since it provides free hosting and can be used to maintain and continuously update the application while it is deployed. If more resources are made are available and a more permanent solution is needed, I would consider using Amazon Web Services for hosting.