Let’s build an app that will act as a ***frontend client*** for a social media platform

General guidelines

1. Make sure to properly organize your code.
2. Make sure to not repeat code if it can be avoided.
3. Remember ***decoupling***.
4. Make sure you understand your own solutions and the decisions you made in order to build it.
5. Make sure to properly document and format your code.

Reminders

1. Remember our rules about *wiring* the html, css and javascript using selectors, tag names and tag attributes.
2. Remember CSS validation and its “limited usefulness”.
3. Feel free to use any layouting tools you may need. Examples - tables, ordered list. You can also position the elements based on their natural behaviour by using pure css rules. Remember the purpose of HTML(structure, skeleton, data) and CSS(layouting, aesthetics).
4. Remember that input elements are just for that. Getting input from the user. Look through the UI, and think what data am i requesting the user to give me? What do i want to do with that data in the future? What will be the correct way to organize it(think in tables and lists)?

Specification and requirements

The application should be able to:

1. Display an endless feed/stream of items/posts. We will call it *the main feed*. Think about the main content that users are allowed to post. Links from youtube, links to music stream or a music file, video files, long text content, short text content, maybe link to other users, etc… Think about how you want to *model* the different types of user posts.
2. Think about the different user roles in your platform. For example, admin user(which is able to do anything and see everything), a regular user, anonymous user.
3. Users should be able to sort the items/posts on the main feed by date(ascending/descending).
4. Allow easy and circular navigation between different parts of the application.
5. Users should be able to register.
6. Users should be able to delete their account.
7. Users should be able to get notifications about new items.
8. Each user will have its own profile page.

9. Each user should be able to manage its own profile and add content to it(think about your profile is other social media platforms).
10. Each user should be able to access other users profiles.
11. Each user should be able to decide which users can access its profile(everyone/public, only friends/private). Pay attention to details. What will be shown in each case? How will the functionality of the profile page change in each case(case 1 - the owner of the profile, case 2 - someone who is not the owner but allowed to view the profile, case 3 - someone who is not the owner and is not allowed to view)?
12. Users should be able to create new items/posts.
13. Users should be able to view all their own created posts.
14. Users should be able to request to be a friend of other users.
15. Users should be able to decide if they want to approve or decline the requests.
16. Users should be able to see all their friends.
17. Users should be able to unfriend a user.
18. Users should be able to comment on their own posts, and other posts of other users.
19. Each post should have comments.
20. Users should be able to comment on a comment.
21. Each item/post/comment should be `*likeable*` by all the users who are allowed to view it.
22. Each user should be able to know which users liked its posts/comments.
23. **All** input must be validated on the frontend in a meaningful way(we do not have to use javascript for that).
24. The frontend must be visually appealing(same “look and feel” throughout the application’s UI. feel free to use frameworks).
25. Use localStorage to store data for each client(will be kept as long as the user does not clear its browser’s cache).
26. Use json server(as shown in class, see pdf on how to install npm and node on windows) for your mock backend.
    a. Mock means fake. Mimicking the behaviour of something real.
    b. We only care that the server sends us the data in the agreed structure, we do not care how the server does it(the server should worry about that).
    c. We can use fake data in order to build and test our GUI(Graphical user interface) in a way which is ***decoupled*** from the actual server.
    d. Here is a handy utility to generate fake data.

Application flow

- Think about what should happen after each user action.
    - What should the user see after login?
    - What should the user see after posting a post?
    - What type of notifications will there be? And how will they be displayed for the user?
    - What happens if there are errors? (Server is down, user tries to do something that the user is not allowed to do)

Code base

- Your project should be hosted on github or bitbucket
- Make sure you set your git config info to your full name and email
- Make sure to include a README.md file which includes - your full name, course name and number(full stack 4580), the title of the assignment(Full stack project - client - version 2), A short description of the project and instructions on how to run it.
- Make sure to create a tag commit for the first version.
- Make sure to create a dev branch, and various feature branches while developing.