Timelines Project Part IV

Yue Zhang and Jeannette Pepin

Table of Contents

1. What features were implemented?

We implemented

- UR-01: Edit profile
- UR-02: login/log out
- UR-03: post browsing
- UR-04: change privacy of post
- UR-05: post creation
- UR-06: add tags to post
- UR-12: choose a nickname to display
- UR-13: View posts of other user in Feeds tab
- FR-07: allow multiple tags on a post
- FR-08: offline application functionality (with models serialized client-side into a cookie)
- FR-10: Allow viewing of public posts before logging in
- NR-01: User friendly
- NR-02: System is responsive to mobile or desktop
- NR-05: Each user can only manage the information on their own account
- NR-10: The system should load entries in user feed within 2 seconds

The following use cases have been completed:

- UC-003: User can make a text post and add tags without image
- UC-007: User can view posts on their dashboard by tag
- UC-015: User can edit their profile and choose a nickname

2. Which features were not implemented from Part 2?

Most of the features from Part 2 were not implemented -- we do not have admin functionality and we do not generate graphs in the statistics page. We also do not have the friends, post editing, or post deletion features.

3. Show your Part 2 class diagram and your final class diagram. What changed?

The following is our Part 2 class diagram and final diagram:

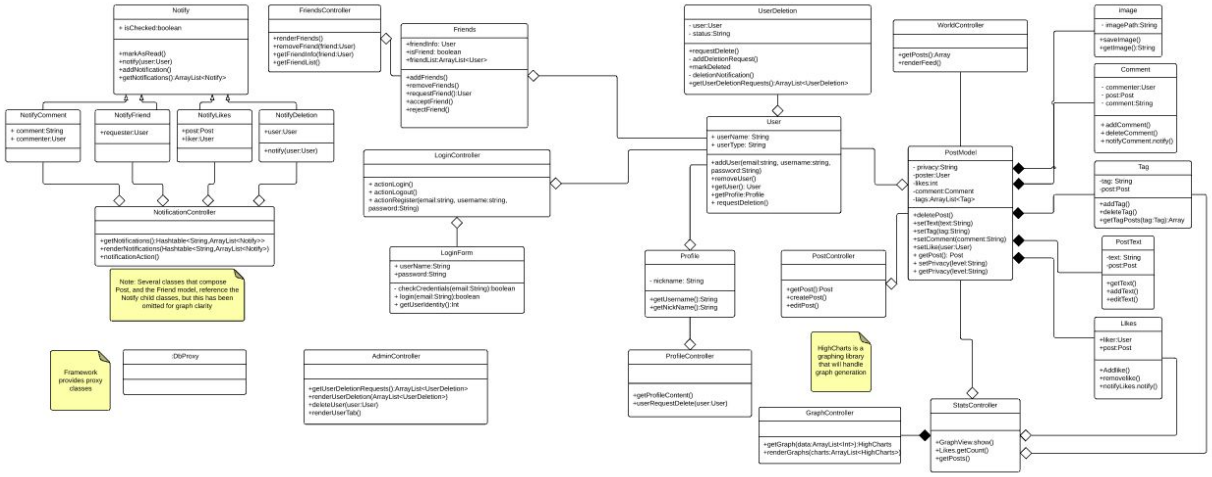

Fig 1. Part 2 diagram

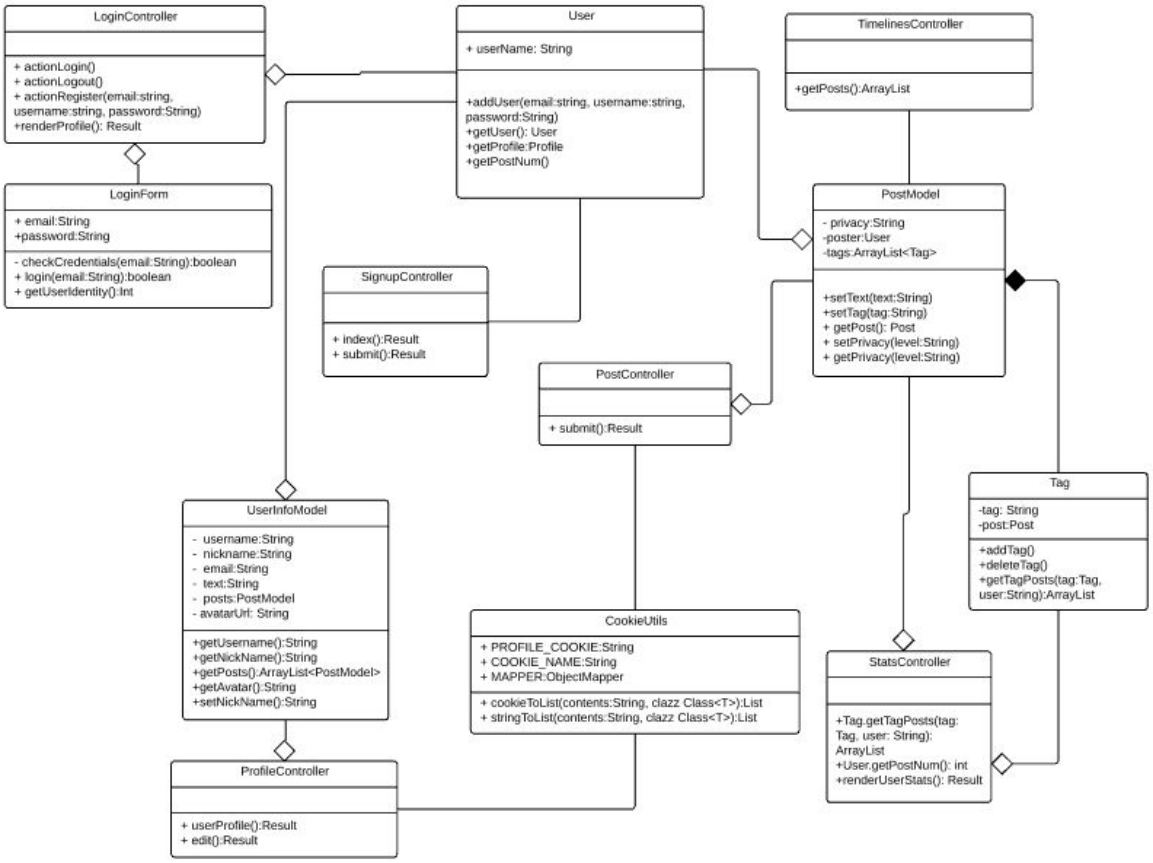

Fig 2. Final class diagram

3a. Why? If it did not change much, then discuss how doing the design up front helped in the development.

The final class diagram has not changed much in nature, but it's much smaller due to several factors. Our third teammate dropped the course right after the completion of Part 2, so our velocity took a hit. We ended up spending most of our time stuck on details of configuration and implementation, such as installing and setting up the framework across different development environments, database setup, and practical front-end web implementation details that required a lot of research (neither of us have a strong background in web technologies). Having the design laid out from the start aided in making the progress that we did, and using the MVC architecture pattern enabled us to work in parallel on the front and back end of the application.

The implementation of the offline browsing and post creation feature lead to the addition of one new class (CookieUtils) for serializing model data, and the modification of several controllers and models to work with this, which we did not originally plan for. This implementation would be a candidate for refactoring, since it was not included in the original design, and was added somewhat organically.

4. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?

Our final implementation did not make it to the features where we would have used certain design patterns, such as Observer for the notifications.
There are several design patterns that we did not consider initially, but would likely be good fits for parts of the project. We could implement the Observer design pattern to make the Stats page render new posts and update the number of posts of the current user dynamically. We could apply the Decorator design pattern if we were to implement the ability to make a video or image post in addition to having text on a post. We could also have used the Strategy design pattern in the Stats page if we had been able to implement viewing different types of aggregation graphs based on the user data.

5. What have you learned about the process of analysis and design now that you have stepped through the process to create, design and implement a system?

Analysis and design of the system -- especially the mockups and user requirements -- aided in getting the group on the same page for what we needed to build, and allowed us to complete some work in parallel by designing around loose coupling and the MVC architecture pattern. However, the implementation demonstrated that design undertaken without sufficient knowledge of the problem domain will be optimistic in scope at best, and at worst inherently flawed. We spent a large amount of time stuck on trivial problems or pursuing dead ends. While the original design plan did aid in planning and guiding our work, the ramp-up period of learning the framework and installing tools and dependencies accounted for most of the time we had to work on the project. A certain amount of struggle with the learning curve of a new technology seems

to be fairly standard in software engineering, and while we didn't make as much progress as expected, the design did ensure that it wouldn't be hard to extend our work if we had had more time.