Peeved

Timeline

- Monday - P3 pitch, back-end, DB connect, models/Schemas, server, controllers, Routes, JWT auth, test all CRUD
- Tuesday - services/config, axios, Routes, Nav, Registration, Login, Home, New-post, Edit-Post, render
- Wednesday (MVP Due) - screens layout/grid, mobile responsiveness, deploy
- Thursday (post MVP) - post likes, posts search, comments, hamburger menu, dropdown/pop-up login, footer
- Friday - additional styling animations
- Monday (9AM Check Ins; Project Presentations at 1PM)

Team

Abdi Osman, John Tran, & Rick Hertel (Git Maintainer)

Personal Strengths

Abdi - Styling and Design
John - Front end
Rick - Backend, front end structure and routing

Personal Challenges (Could Use More Exposure)

Abdi - Back-end connection to the front-end
John - Authorization
Rick - Front-end functional components handling / rendering

GROUP PLANNING

Team Goals & Values

- Interactive functional application with non-buggy code
- User friendly, interactive UX, UI
- Strong team communication
- Achieve milestones, stick to timelines

Team Communication Preferences

- Slack at any time
- Daily team stand-ups, after squad stand-up, to go over goals for the day and execution
- Hourly Zoom check-ins during the day (must check in with team mates if stuck in one spot >20-30min)
- Near immediate response during class-time
- Outside of class, reasonably respond when possible, life permitting
- Set/attend after hours sessions at agreed upon times of the team

CODING PRACTICES

Conventions:

Branch Naming

- master, development branches
- sub-branches use "initial-feature-dev" convention, ie: *ao-register-dev*

Practices (Please note, this is an example. You don't need to follow it to a T, but it will prevent further frustrations.)

1. **Never** code on, merge, or push to Main
2. Git Maintainer will create a Development branch and set the upstream so it's available to all group members.
3. Group members should run a *git checkout -b dev* and a *git pull origin dev* to link their remote branch to the origin branch.
4. Group members can then run *git checkout -b feature-branch-name* from their local Development branch.
5. Make code additions.

6. If there are no conflicts, you can initiate a pull request on GitHub. Confirm the PR is pulling from your branch and comparing to dev; title the PR “BRANCH NAME to dev.” This PR and merge will require the git maintair’s approval to finalize the merge.
7. Once that merge is finalized, all group members should add and commit their code to their local branch.
8. Then, they should run a *checkout* to their local Development branch, followed by a *git pull origin dev*;
9. Then they should run a checkout to the feature branch they were working on and also run a *git pull origin dev*.
10. Continue coding on your feature branch. Return to step 6 as needed.
11. Once your Development branch is ready to merge with Master, the Git Maintainer can make a PR from Development to Main. This PR requires at least 3 of 4 group members’ approval to finalize the merge.
12. Once that merge is finalized, all members should *pull origin master* to all local main and sub branches, if possible.

Task Tracking
Trello Board

MongoDB Database
Express Routes
Collection Names/Schemas
React Component Names & Folders