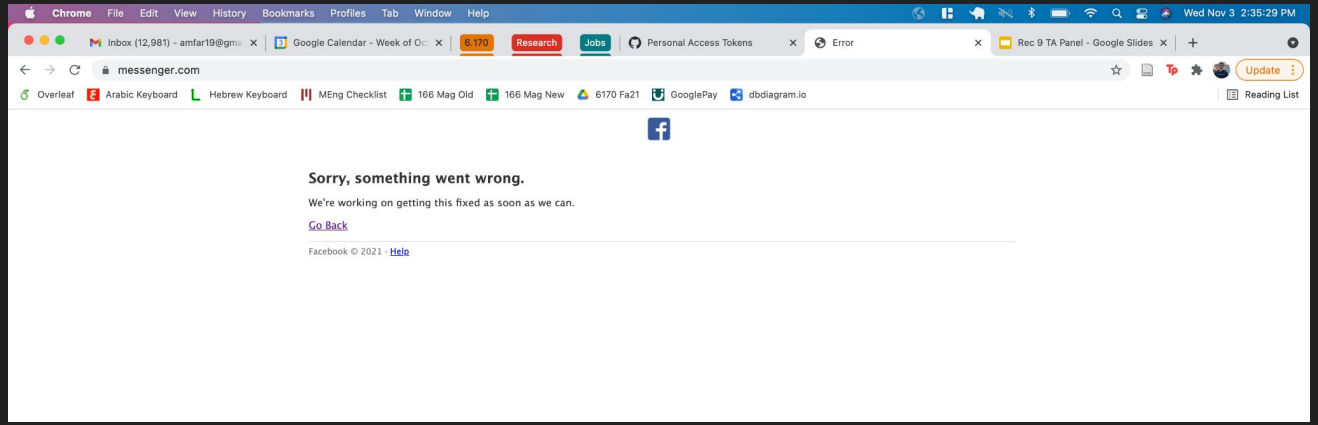
6.170 TA Panel



Agenda

1. Final Project Experiences and Advice
    a. Suggested Workflow
    b. Team Dynamics Tips
    c. Design First
2. Q&A

Suggested Workflow

- How to partition the work?
  - **Design** -- ideation, concepts, models, sets & relations
  - **Implementation**
    - *Backend* routes, middleware, data persistence / DB
    - *Frontend* components, layout, styling, data storage & patterns
    - Git VCS workflow -- pull-requests, branches, tagging
- How would you split this up among your team?

Suggested Workflow

- How to partition the work?
  - **Design** -- ideation, concepts, models, sets & relations
  - **Implementation**
    - *Backend* routes, middleware, data persistence / DB
    - *Frontend* components, layout, styling, data storage & patterns
    - Git VCS workflow -- pull-requests, branches, tagging
- Suggested workflows:
  - Breaking up the work by concepts & features: each member devotes their efforts to the design and implementation of a concept(s)

Suggested Workflow

- How to partition the work?
  - **Design** -- ideation, concepts, models, sets & relations
  - **Implementation**
    - *Backend* routes, middleware, data persistence / DB
    - *Frontend* components, layout, styling, data storage & patterns
    - Git VCS workflow -- pull-requests, branches, tagging
- Suggested workflows:
  - Breaking up the work by concepts & features: each member devotes their efforts to the design and implementation of a concept(s)
  - Frontend & backend separation: Alyssa P. Hacker builds out backend for upvote concept, Ben Bitdiddle builds out frontend for it.

Suggested Workflow

- How to partition the work?
  - **Design** -- ideation, concepts, models, sets & relations
  - **Implementation**
    - *Backend* routes, middleware, data persistence / DB
    - *Frontend* components, layout, styling, data storage & patterns
    - Git VCS workflow -- pull-requests, branches, tagging
- Suggested workflows:
  - Breaking up the work by concepts & features: each member devotes their efforts to the design and implementation of a concept(s)
  - Frontend & backend separation: Alyssa P. Hacker builds out backend for upvote concept, Ben Bitdiddle builds out frontend for it.
  - Optimize for what people want to learn, not what people are experienced with. For each area, identify the “subject matter expert(s)” and the “developer(s)” in that area. Developers do the work, and ask for guidance (when needed) from the subject matter exports

Team Dynamics Tips

- Have internal soft deadlines for small deliverables.
- You should also be checking semi regularly that the different pieces of code and features work together as expected.

Team Dynamics Tips

- Have internal soft deadlines for small deliverables.
- You should also be checking semi regularly that the different pieces of code and features work together as expected.
- Choose someone to be a sort of project manager to help keep everyone accountable about their deliverables.
- Communicate clearly and early if things come up that prevent you from delivering a piece of code you signed up for.

Team Dynamics Tips Part 2

- Possibly have a system for tracking progress like Asana or something else.
- Create as many reusable components as possible for consistent styling and so you don’t have a lot of repeated code.

Team Dynamics Tips Pt 2

- Possibly have a system for tracking progress like Asana or something else.
- Create as many reusable components as possible for consistent styling and so you don’t have a lot of repeated code.
- Have synchronous meetings so everyone is on the same page
- It's helpful to work at the same time, on a Zoom / Discord call
    - Can ask questions / coordinate
    - Can help each other debug via screenshare / VSCode liveshare

Design First

- Don't start fleshing out implementation until you establish your design
  - makes it easier to collaborate -- even if some pieces aren't implemented yet, you know what will be available to you
  - much harder if you need to rework code

Design First

- Don't start fleshing out implementation until you establish your design
  - makes it easier to collaborate -- even if some pieces aren't implemented yet, you know what will be available to you
  - much harder if you need to rework code
- Design includes:
  - Resources (models, database tables / keys)
  - Functionality (routes)
  - Structure (frontend components and flow)

Design First

- Don't start fleshing out implementation until you establish your design
  - makes it easier to collaborate -- even if some pieces aren't implemented yet, you know what will be available to you
  - much harder if you need to rework code
- Design includes:
  - Resources (models, database tables / keys)
  - Functionality (routes)
  - Structure (frontend components and flow)
- Quick prototypes / proofs of concept with some hard-coding is good if you want to try something out
  - This should be code you are okay with throwing out if it doesn't work
  - Stub out unfinished functions

Resources

- Online Resources Doc with general links to materials from the class, deployment, etc
- Final Project Programming Resources Doc with links to VCS workflow, visual design, map resources, vue ecosystem packages, and different databases
- Next Wednesday, TAs will be showing you how to employ data-preloading on your web server and how to use MongoDB for data persistence
- Your assigned TA!

Q&A

- We will put this presentation on canvas

What questions do you all have?
Let’s chat :)