Discussion of routes

- GET party/users/username
    - Get rid of party/
    - Separate route in users that is GET users/username to get all parties for that user
- Claiming an item
    - Put userID of who is claiming item in request body
    - PUT /party/partyId/supplies/itemId
- RSVP
    - PUT /partyId/guests with userId in request body

Discussion of user authentication

- Must tie invited user to an email or phone number to ensure that they are the correct user
- MVP pick one of email or phone number
- Two cases when email is invited
    1. Already has account
        a. Can query for that user and update accordingly
    2. Does not have account
        a. Redirect user to create an account
        b. Can make them an account on the spot, user logs in to change password

App.jsx

- Outermost component of app
- Fritter
    - Different pages that all stored some subset of tweets
    - So stored list of tweets and populated that list differently
- Since we have one page, may not need functions in App.jsx
- Can store most functionality in Homepage.jsx
- partyServices are not react code, just communication with server
    - Look at promises
- Look at clientRoutes.jsx
- Nav bar goes here

Validating emails

- No good regex
- Send verification emails
- Not necessary for MVP, good for final product

Steps remaining for MVP

- User's invited see party
- Host update guest list
- Sign up for and add items

- Edit supply list
- Delete old Fritter stuff

Summary of discussions and advice:

1. Make routes RESTful: Use POST for creating new objects and PUT for updating one object, so RSVP for a user and claiming item on supply list should both use PUT. URL describes item to modify and request body contains information used to modify that item.
2. User authentication: Invite users to event by email. Query users by email and if found, user already has an account, so add party to their account. If user not found, create an account for that user and send verification to email. Too hard to validate emails otherwise.
3. App.jsx: It is the outermost component of our app. Things that are consistent across app go here, like a nav bar. We can store most functionality on Homepage.jsx since we have one page per user that displays information based on the same query. Services are not react code, they can work with any framework.
4. Delete fritter code for MVP turn in.

Summary of new decisions:

1. Change routes to make them more RESTful
2. Delete most function in App.jsx and rewrite functions in Homepage.jsx. Rewrite clientRoutes.jsx to match our app.
3. Delete Fritter code.

Changes to problem analysis or design:

1. Add email field to user registration