Report: HTTP Requests / Routes

In the report, include a list of HTTP routes implemented in the server, and the mock server method that they replace. For each, briefly describe:

- The *verb*, *target*, and *body* of the request.
    - Example: `POST /feeditem [feeddata]`, where `[feeddata]` is a JSON object describing a Feed Item.
- What the HTTP request does, and what mock server method it replaces.
- What resources the HTTP request creates/modifies/etc.
    - Examples: “Updates `/user/:userid`”, “Creates a new `/user`“…
- Who is authorized to use the request
    - Example for `PUT /user/:userid userdata`: “A user with the specified `:userid` can issue this request. Administrators can also make this change on any `:userid`.”
    - Example for `POST /feeditem feeddata`: “The body of the HTTP request contains an “author” field containing a user ID. The requester must have the same user ID.”
    - If you have a hard time explaining it generally, feel free to include examples. Example: “A user with ID 4 can issue a `PUT /user/4`, but a user with a different ID cannot.”

Report: Special Server Setup Procedure

If your server has any advanced features that require additional setup besides `npm install` and `node src/server.js`, include a section in the report that describes how to set up these features. Include details on how to tell if the feature is working properly.

Report: Individual Contributions

Include a section that describes each startup founder’s contribution. Each startup founder should be responsible for at least one product feature and its HTTP route(s). Name the feature and the HTTP routes that the founder implemented / was responsible for.

Report: Lingering Bugs / Issues / Dropped Features

If your application has any lingering bugs, issues, or missing/dropped features, include a section listing these.

1: HTTP Requests / Routes

getUser GET/user/:userid

{_id: number
Name: string
Bio: string
}

Returns the id, name, and bio of the user that has the userid

Replaces getThreads GET /user/:id/messages

[thread]
Returns {:threads} for the user, representing all the threads a user is a part of
Requires user to be logged in

Replaces sendMessage PUT /user/:userID/messages/:threadID - Saul

{message}
This add the message to the thread specified by threadID, and set's the author to be the index of userID in the thread
Requires user to be logged in and a participent of the thread

Replaces getOrCreateThread POST /user/:userID/messages/:

{otherUserID:}
This gets the thread between userID and otherUserID or creates it if it doesn't exist. It returns {id} where id is the new thread id.
Requires userID to be logged in

getAllPost GET /posts/ where posts are all the posts users have created. - Kazi

getUsersPosts GET /user/:id/posts - John

getPost GET /post/:id - Avery

createPost POST /post/ - Huy

Requires post, which contains an authorID, name, description, and tags, which has and requires the schema from post.json. Returns a new post with the authorID, name, description tags, and the generated id and date.

updatePost PUT /post/:id where a user with the specified id can edit his/her post. - Hana

getTags GET /tags/ where tags are received from a post. - Evan

2: Special Server Setup Procedure

To view our website “npm run serve” in the “/client” directory is required to be run along with “node src/server.js” in the “/server” directory and the URL is “http:///localhost:8080/” due to visibility. This has to be done because currently everything isn’t viewable through the server alone, which is why “http://localhost:3000/” will say “cannot GET /” instead of showing our homepage.

a.

3: Individual Contribution

Kazi Ahmed: Created, initialized and edited client folder. Edited the package.json so the client can do what it needs to. Wrote Special Server Setup Procedure part a based on completed work. Fixed a small syntax error so “node src/server.js” works.

Huy Nguyen: added the xhr info to client/app/server.js, created function for getUser and createPost for client and server.

Jonathan Shepherd: wrote functions for get users posts on client and server.

Saul shanabrook: setup the server initially, added babel polyfill so we can user Object.values on the server,moved over messaging methods.

4: Lingering Bugs / Issues / Dropped Features

Bug: We have not added authentication for any routes.