CCSGA Comments: API for the Messaging Service

- Create
    - Initiate new conversation
        - POST /api/conversations/create
        - Request payload: {“revealIdentity”: boolean, “messageBody”: string, “labels”: string[]}
        - Response payload: 201 with {“conversationId”: number, “messageId”: number}, 401 if not authenticated, or 403 if banned
    - Reply to a conversation
        - POST /api/conversations/<conversationId>/messages/create
        - Request payload: {“messageBody”: string}
        - Response payload: 201 with {“messageId”: number}, 404 if not found, 401 if not authenticated, or 403 if banned or if not authorized to reply to this conversation
    - Add an admin (side effect: un-bans)
        - POST /api/admins/create
        - Request payload: {“newAdmin”: string}
        - Response: 201 if successful, 200 if newAdmin was already an admin, 401 if not signed in, 403 if requester is not signed in as an admin
    - Add a CCSGA rep (side effect: un-bans)
        - POST /api/ccsga_reps/create
        - Request payload: {“newCCSGA”: string}
        - Response: same codes as for adding admins
    - Ban a user (side effect: removes admin and CCSGA roles)
        - POST /api/banned_users/create
        - Request payload: {“userToBan”: string}
        - Response: same codes as for adding admins
- Read
    - Get authenticated user’s info
        - GET /api/authenticate
        - Response payload: 200 with {“isSignedIn”: boolean, “username”: string | “not signed in”, “displayName”: string | “not signed in”, “isBanned”: boolean, “isCCSGA”: boolean, “isAdmin”: boolean}
    - Get user’s conversations
        - GET /api/conversations
        - Response payload: 200 with {<conversationId>: same object as GET /api/conversations/<conversationId> (with as many such k/v pairs as conversations the user has)}, 401 if not authenticated, or 403 if banned
    - Get the messages and other details of a single conversation
        - GET /api/conversations/<conversationId>(?overrideAnonymity=true -- optional)
        - Response payload: 200 with {

- "messages": {<messageId>: same object as GET /api/conversations/<conversationId>/messages/<messageId> (with as many such k/v pairs as messages in convo)},
- "status": string,
- "labels": string[],
- "isArchived": boolean,
- "allIdentitiesRevealed": boolean,
- "ownIdentityRevealed": boolean,
- "isRead": boolean // whole conversation is read

■ }, 401 if not authenticated, 404 if not found, or 403 if not authorized/banned

○ Get the details of a single message[1]

■ *GET /api/conversations/<conversationId>/messages/<messageId>(?overrideAnonymity=true -- optional) —* ***iceboxed, but left here in case users need to be able to share individual message links***

■ Response payload: 200 with {

- "sender": {
  - "username": "anonymous" | string,
  - "displayName": "Anonymous" | string
- }, "body": string,
- "dateTime": string,
- "isRead": boolean

■ }, 401 if not authenticated, 404 if not found, or 403 if not authorized/banned

○ Get list of admins

■ GET /api/admins

■ Response payload: 200 with {"admins":

- [{
  - "username": string,
  - "displayName": string,
  - "isBanned": boolean,
  - "isCCSGA": boolean,
  - "isAdmin": boolean
- } (object repeated for each admin)]

■ }, 401 if not signed in, or 403 if requester is not signed in as an admin

○ Get list of CCSGA reps

■ GET /api/ccsga_reps

■ Response payload: 200 with {"ccsgaReps": [same object as inside GET /api/admins, repeated once for each rep]}, 401 if not signed in, or 403 if requester is not signed in as an admin

○ Get list of banned users

■ GET /api/banned_users

[1] Grey highlighting indicates iceboxed routes and fields that are not currently implemented in the backend.

        - Response payload: 200 with {“bannedUsers”: [same object as inside GET /api/admins, repeated once for each banned user]}, 401 if not signed in, or 403 if requester is not signed in as an admin
- Update
    - Archive, change the status of, change the labels of, or deanonymize (for oneself; see GET request for the admin feature) a conversation. Can also update the read/unread state of a conversation (i.e., either mark the most recent message as unread, or mark all of the messages as read)
        - // mark as read/unread/archiving/unarchiving happens just for that individual user
        - PATCH /api/conversations/<conversationId>
        - Request payload: {“revealIdentity”: true | null, “setArchived”: boolean?[2], “setStatus”: string? ~~(or perhaps status id?)~~, “setLabels”: string[]?, “setRead”: boolean?}
        - Response payload: 200 if successful, 401 if not authenticated, 404 if not found, or 403 if not authorized/banned
    - Update the read/unread state of a message
        - PATCH /api/conversations/<conversationId>/messages/<messageId>
        - Request payload: {“setRead”: boolean?}
            - // mark as read/unread/archiving/unarchiving happens just for that individual user
        - Response payload: 200 if successful, 401 if not authenticated, 404 if not found, or 403 if not authorized/banned
- Delete
    - Remove an admin
        - DELETE /api/admins/<username>
        - Response: 200 if successful, 401 if not signed in, 404 if username was already not an admin, 400 if request was cancelled because it would have removed the last admin, 403 if requester is not signed in as an admin
    - Remove a CCSGA rep
        - DELETE /api/ccsga_reps/<username>
        - Response: 200 if successful, 401 if not signed in, 404 if username was already not a rep, 403 if requester is not signed in as an admin
        -
    - Unban a user
        - DELETE /api/banned_users/<username>
        - Response: 200 if successful, 401 if not signed in, 404 if username was already not banned, 403 if requester is not signed in as an admin
    - // TODO: perhaps CCSGA reps deleting conversations without actual backend deletion)

[2] `?` indicates an optional property.