Report

HTTP Requests / Routes

1) **POST /feedback [feedbackdata]**, where [feedbackdata] is a JSON object describing a Feedback.

POST /feedback [feedbackdata JSON object] creates a new Feedback with the specified data. The server can tell the client what the new Feedback's ID is in the response. It replaces postFeedbackData method in mock server.

Creates a new /feedback. (when the user clicks on "Submit Feedback")

The body of the HTTP request contains an "author" field containing a user ID. The requester must have the same user ID.

2) **POST /searchpeople**

POST /searchpeople searches for users using query ('/searchpeople?searchTerm=' + queryText). It goes through the whole user collection and filter by users' names. If a name matches, it shows up on the people profile page. If no query is typed in, the search returns all users.

SearchQuery (created by Anh Duc Bui) checks if query is undefined and returns PeopleProfile.

3) POST /interviewsession [interviewitem], where [interviewitem] is a JSON object describing a Intervewsession.

POST /interviewsession [interviewitem JSON object] creates a new Interviewsession with the specified data. The server can tell the client what the new Interviewsession's ID is in the response. It replaces postInterviewSession method in mock server.

Creates a new /interviewsession.

4) GET /user/:userid/interviews

This gets all the interviews that of the user (an authorized user can view only their interview history).

5) GET /interview/:interviewId

Get the user and problems object in the database with a given interviewId. Showing the interview questions in interviews.js

6) POST /interview/:interviewId

POST /interview/:interviewId creates a new interviewSession and updates the database with it given the user's (interviewer's) id. When the user clicks "Find me an interview" on the matching page, it leads to this route.

The mock server method that was replaced by this route is postInterviewSession.

Only the current logged in user is authorized to issue such request.tgt

7) POST /resetdb

Similar to the Facebook Reset Database route.

8) GET /user

Does not require authentication. Get all users' profile.

9) GET /user/:userid

Get user information with that id, for userprofile.js

Individual Contributions

- Ye Sung (Rebecca) Kim:
  - Feedback.js: Migrate/Add a route to the server for postFeedbackData
- Anh Bui:
  - Fixed overall bugs.
  - SearchQuery.js
  - Homepage.js
- Ngan (Sylvia) Hoang:
  - Matching.js, migrated code to the server
- Thanh Pham:
  - Interview.js: migrate getInterviewSession to server
- Tri Nguyen:
  - History.js: migrate getInterviewData to server.
  - Add the Firebase Code Editor for two people to collaborate on it.
- Tien Dao:
  - Peopleprofile.js, userprofile.js: migrate searchForUsers to server
  - Fix user profile CSS bug from last submission

Lingering Bugs / Issues / Dropped Features

_ When the server matches users for interviews, it doesn't care whether the user is currently in the matching pool or not. This feature will be implemented when we have a proper user authentication.

_ The date and time in page history is not working properly yet.

_ The number of interviews done does not match with some users' history display (some displays as "You have no interview in your history").
_ Voice chat using WebRTC is still not implemented yet. We are still working on solving this problem.

Honors Section

HTTP Requests / Routes

DELETE /chatsession/:chatid/memberlists/:userid : deleteChatMember(), remove chat member from the chat session authorized only for the current login user (the button is disable when you visit another user's session)
PUT /chatsession/:chatid/memberlists/:userid : addChatMember(), add chat member to the chat session authorized only for the current login user (the button is disable when you visit another user's session)
DELETE /notification/:notificationid : deleteNotification(), delete a notification in the database using the ID, authorized only for the current login user
PUT /notification/:notificationid/status/:status : updateNotificationStatus(), update a notification's status in the database using the ID, authorized only for the current login user
POST /chatmessage : postChatMessage(), post a chat message, authorized only for the current login user checked with the owner of the chat message
GET /chatsession/:chatid : getChatSessions(), get the chat session using the ID, authorized for the current login user
POST /notification : postNotification(), post a new notification, authorized for the requester of the notification
GET /user/:userid/online : getOnlineUsers(), get a list of online users authorized for the current login user
GET /user/:userid/nearby : getNearbyUsers(), get a list of user nearby (within a default radius 10000), authorized for the current login user
GET /user/:userid/notification : getNotifications(), get notifications that the user needs to see (using the requestee field in the notification), authorized for the user with that userid.

- Now the notifications when you click checkmark, you can get added to that user's chatSession, and then that checkmark turn into a button that can navigate you to that user's chat room.
- Now you can remove yourself from the chat room that you join via the notification by pressing the X button.
- Now you can edit the members in your chat room through that "Edit Member" button in the room

Lingering Bugs / Issues / Dropped Features

_ the app doesn’t get the user locations yet. It will be implemented when we do the login page. The users’ location will be updated whenever the user logs on.