Team Kiwi

Startup Assignment 6

Using 1 late day.

<u>Report: HTTP Requests / Routes</u>

HTTP requests used by Academic Page and Service Page

GET /user/1/feed/type

where type is either academic or service. This request simply returns the feed of the user to see. The user 1 can use it in this case. It is modified from getFeedData in app/server.

POST /feeditem/type

where the type is either academic or service. This request lets the user post either an academic or service post in the feed. The current user can creates a new feeditem and add it to the feed in the server. It is modified from postStatusUpdate in app/server.

DELETE /user/userId/feed/type/feeditemId

where the user ID is the current user's ID and the feeditem ID is the ID of the feeditem that the user wants to delete. This request lets the current user delete an feeditem from the feed based on the ID. It is modified from deleteFeed in app/server.

POST /search/queryText

where the queryText is the string that the current user entered in the search bar. Then it will post the feedItem that contains the string entered by the user. It is modified from searchForFeedItems in app/server

PUT /feedItem/feedItemId/likeList/userId

where the feedItemId is the feeditem to like, the listList shows how many usersLike this post and the userId is the user that is liking the post. By clicking the like icon, the user with the userID likes the feedItem with the feedItemId. It is modified from likeFeedItems.

DELETE /feedItem/feedItemId/likeList/userId

where the feedItemId is the feeditem to unlike, and it removes the user with userId from the feedItemId's likelist. It is modified from unlikeFeedItems in app/server. .

PUT /feedItem/feedItemId

where the feedItemId is the feedItem being viewed by the current user. It is modified by the likeFeedItem in app/server

GET /comment/:commentid/:userid

where the commentid is the id that comment has in the database and userid is the id of the user and the userid is primarily using for authentication. This will return a comment for a given feed

POST /feed/:feeditemid/comment/:userid
feeditemid is the id of feed that a user is currently posting comment, this function will create a new comment and store it in the database with need id.

HTTP requests used by Profile Page

GET /user/1/profile/userId
where the userId's profile is being get. The current user can use it and it is modified by the getUserData in app/server.

PUT /user/userId/profile
where the current user can get the profile. It is only accessible by the current user. And it is modified from getUserData in app/server

HTTP requests used by Schedule Page

POST /schedule/
where the schedule is posted by the current user with its attributes. It is created in postSchedule in app/server.

GET /schedule/:userid
where userid is the current user Id corresponding to the database. It is modified from getScheduleData in app/server.

DELETE /schedule/:userid/:scheduleid
where the userid is the current user and scheduleid is the id of that schedule you want to delete. It is modified from deleteSchedule in app/server.

HTTP requests used by Configuration Page

GET /config/user/ setting
where setting is the username, password, and email of the user. This request does not change anything, just show the certain attributes of user/userid/profile. The user who is currently using this page can use this. It is modified from getUserSetting in app/server.

PUT /config/user/ setting
where setting is the username, password, and email of the user. It modifies certain attributes of user/userid/profile if the user choose to change it. The user who is currently using this page can use this. It modifies updateUserSetting in app/server.

HTTP requests used by Message Page

GET /messagebox/boxId/participantlist/
where the messagebox is created by the user and the participants are the people in the message. By using this function you can access the profile page of the other users in the list. It is created in getParticipantProfiles in app/server.

GET /messagebox/box_msg_id
where the box_msg_id is the ID of the message box that the user want to get, the current user can access it and it is created in getMessageBoxServer in app/server.

POST /messagebox/:box_msg_id/send/:user_id
is used to send a message that the user typed in the message box.

GET /users/:userid/recentmsgboxes/:numberofboxes
is used to get the user's a number of recent message box ids that he or she have joined.

PUT /messagebox/create/:user_id
is used to create a new message box when the user hits 'New Messagebox' in Message page.

PUT /messagebox/:box_msg_id/add/:user_id
is used add a participant into a message box.

Report: Special Server Setup Procedure

Our group does not have any special server setup procedures, although we do have a textbox that lets people change the users by their ID in the database for Mockup Purposes. Just simply enter a number from 1 to 6 and click change user button below to do it.

Report: Individual Contributions

Thien: He implemented server's routes that handle XHR requests from clients when a user interact with the web-app in the Message page. In the Message page, when the user clicks "New Conversation", a new Message box is created, which allows the user to have a new conversation with his or her friends. To chat with a new person in a conversation box, the user has to click the button "Add Participant", which will show a dialog asking for the participant's id. All the data passing from the client to the server and vice versa in the Message page are done via XHR calls from the clients and handled by Express routes, both of which were implemented by Thien.

Jucong: Finish detail pages. User now able to post feed with selected category and able to view comment of given feed. Also, when user click the cross on the upper right corner, the feed will disappear and remove from the list_of_feeditems for the given user. If the user is the author of the feed, the feed will also remove form feed item in the database.

Karen: Created profile page for showing user info from database. Created server functions to get user info, display info, and update info.

Tim : Created search first without the server then moved it into the server as an http post call. Created the error banner and also handled authorization in the server.Search currently only returns academic feed.Fixed responsive UI bugs in service and academic home pages. I'm working on returning both academic and service feeds but there seems to be a problem with calling read Document twice in a row.

Xin: Created "add to schedule" dialogue for adding schedule. Implemented Created schedule page for showing data from database, and implemented "add to schedule" function as well as the server functions. Implemented GET, POST and DELETE schedule data from the schedule page in HTTP. Created server functions to do these.

JianYi: Created server functions for viewing the configuration page and edit the configuration page. Also created config.json in order to make it work. Also wrote the HTTP routes and requests.

Report: Lingering Bugs / Issues / Dropped Features

Academic Page:
- None

Service Page:
- None

Message Page:
- 'Enter to Send' button is still not working.
- 'Add Appointment' button is not implemented and is likely to be dropped.
- Recent Message Box does not support scrollable when the user create too much Message boxes.

Schedule Page:

After deleting schedule, need to refresh page manually.

Configuration Page:

None.

Profile Page:

None.