Server functions replaced	***HTML paths***	***Functionality***	***Who can use it***
Get thread	GET /post/:threadid	Retrieves Thread information, so threads may be posted to some component	Any Component
Get full thread	GET /thread/:threadid	Retrieves the entire thread, along with properly indented comments	The thread & replies components use this. It is used to retrieve all info after a reply is made
Get feed data	GET /feed/:userid	Retrieve feed data for a user	User that matches user id. Any user can access the feed data of user 2
Get Board Content	GET /board/:boardid	Retrieves the data for a single board. Data is made up of board information, as well as the OP of each thread.	Anyone
Get pinned posts	GET /user/:userid/pinnedposts	Retrieves information about pinned posts	Only a user can access their pinned posts
Add pinned post	PUT /user/:userid/pinnedposts/:threadid	Adds a pinned post to the users list of pinned posts	Only a user can access their pinned posts
Delete pinned post	DELETE /user/:userid/pinnedposts/:threadid	Deletes a pinned post from a users list of pinned posts	Only a user can access their pinned posts

Get subscribed boards	GET /user/:userid/subscribedboards	Gets the boards a user is subscribed to	Only a user can access their subscribed boards
Get boards	GET /board/	Retrieves all data on all boards	Anyone
Add subscribed boards	PUT user/:userid/subscribedboards/:boardid	Subscribes a user to a board	Only a user can edit their subscribed boards
Delete subscribed boards	DELETE user/:userid/subscribedboards/:boardid	Unsubscribed a user to a board	Only a user can edit their subscribed boards
Get conversations	GET user/:userid/conversation	Gets the conversations a user is a part of	Only a user can get their conversations
Get conversation	GET user/:userid/conversation/:conversationid	Gets a conversation of a user	Only a user can get their conversations
Post message	POST user/:userid/conversation/:conversationid/	Posts a message to a conversation	Only a user can post a message to their conversations
Post reply	POST /thread/:threadid/replyto/:replyid/	Posts a reply to the OP	Only a user can post a reply
Post replyToReply	POST /thread/:threadid/replyto/:replyid/sub/	Posts a reply to any reply	Only a user can post a reply
Create thread	POST /thread/	Creates a new thread and updates the boards	The body of the HTTP request contains an “author” field containing a user ID. The requester must have the same user ID.

Get user	GET /user/:userid	Gets the users information	Only a user can get the information for their account settings
Update user	PUT /user/:userid	Updates the users information	Only a user can update the information for their account settings
Block user	PUT /user/:id/blockedusers/:userid	Adds a user to the blocked user list	Only a user can add another user to their blocked list
Unblock user	DELETE /user/:id/blockedusers/:userid	Removes a user from the blocked user list	Only a user can unBlock a user that they have blocked

Individual contributions

Evan Reiff worked on create thread functionality (POST /thread/), the error banner, and set up the server side server.

Matt Parke worked on the mainpage functionality, that being retrieving data from the server/database for filling out meta data about boards on the frontpage. The route I wrote was GET /boards/ route. I also did the initial setup of splitting the client and server sides of the project.

John Guttadauro worked on Boards, which involved pulling in data from the server to fill out the page. I also was responsible for fixing bugs in pinned post, as well as some editing of the related server methods. /board/:boardId is the route I created.

Nathan Greenberg worked on the server functions for Messaging/Conversations, SubscribedBoards, and Feeds. These are the /user/:userid/conversation/..., /user/:userid/subscribedboards/..., and /feed/:userid/… routes.

Sabrina Dobson worked on implementing search so it actually searches the descriptions of every thread to find matching keywords, and migrated the search function getSearchData /search. As well as all the pinned post methods. /user/:userid/pinnedposts , /user/:userid/pinnedposts2 , /user/:userid/pinnedposts/:threadid , /user/:userid/pinnedposts/:threadid

Patrick Hickey worked on Threads/posting replies to threads -- this involved writing separate methods for replying to the original post and replying to a reply. Additionally, I had to write a function for retrieving the full thread, and add routes for replies to the OP/replies (/thread/:threadId/replyto and /thread/:threadId/replyto/:replyId/sub respectively). As a side note, the thread page has been restructured and bugs from the previous submission have been fixed; now the user can successfully reply, and a text field is opened when the user clicks the “Reply” button.

Shoshana Massarik implemented the server functions for getUserData, the updateUserData and unBlock. Which are the /user/:userid and /user/:id/blockedusers/:userid routes. The issues from the previous submission have been addressed and there is now user feedback when the user changes a setting on the page.

Lingering Bugs/Issues

Minor problems with Threads:

1) When the user clicks Reply, it opens a textfield for everything in that that reply array instead of opening just one textfield. This is caused by the mapping in the render() method in thread.js -- I need some way to associate rendering the textfield of a specific reply, instead of mapping that to every reply in that array. - Patrick

2) When the user posts a reply to a reply, the page does not automatically rerender. Once text has been entered in the field and the submit button has been clicked, the user must hit the refresh button for the new reply to appear. I believe this error is caused by a communication problem between components, because the message generated is “Uncaught TypeError: _this.setState is not a function”. Currently, I attempt to rerender the whole thread when a reply to a reply is made. - Patrick

3) We must also implement a working ViewCounter for threads. Comments is an easy one -- whenever a comment is made, simply increment thread.commentsNo. We can access the View counter in the same way, but we have yet to decide how we want to handle this counter. Should we increment the counter whenever the thread is viewed, or only when it is viewed once by a given user? For the former approach, simply increment thread.viewsNo whenever someone tries to retrieve a thread. For the latter approach, we would need to add a “viewedList” to thread. Then, if a user accesses a thread for the first time, simply add their name to the viewedList. Finally, we could just display thread.viewedList.length instead of thread.viewsNo.

Minor Problems with Account Settings

1) You can’t delete users from the blocked list