# Middlewares:

App Level Handler - middleware:

AppErrorMiddleware

* On any error at app level, a logger catches the error details of origin, error message, time of occurrence and level of importance.
* On providing an invalid route the error is caught at app-level giving an 404 error response and logging the details.

routeIpLogger

* On accessing any route, a logger logs all the header information and IP addresses of the user.

Route Level handler – middleware:

friendCheck:

* It checks if the user accessing the route is either a friend of the owner of the list or the owner itself. It has the option to prevent access to this route by providing a property “actionOnList” as “true” as body parameter.

Auth:

* It is used to control access to particular route. Only on proper verification of the user’s authentication details will the user be allowed to access the route.

# User Management :

SignUp : A new user can signup with the toDoManager by creating a new account, by providing all the required details.

Login: users can login by using email and password.

Get-all-users : users can get list of all available users that can be further requested to be added as friend

Get-single-user : get details of single user account

Edit-user : edit details of single user account

Delete-User : delete single user account

Forgot-password: this functionality allows the client to get new token that will be valid for 3minutes for accessing edit password page

# ToDo Manager –

createNewList : It allows users to create new list in the toDoManager after login. User will not be able to create a list with the same name as one that is already actively existing in the lists

createNewItem : This allows users to create new items in each list. It also prevents users from making items with same names as it will be redundant

editItem : This allows users to be able to edit items details like item title, item description and due date of the item.

editList : This allows user to edit the list details like list title and list description

deleteItem : this deletes the selected item in a list. There is no reverting.

deletList : this allows the users to delete the list selected. It has to be noted that this deletes the entire content of the list, including all items and there is not reverting

getUserAllLists : This allows users to get all the lists that the selected user owns. This can be used to load lists on selecting a user or loading all the lists of the user on loading page

markItemAsDone : Allows user to mark an item as done.

markItemAsOpen : This allows users to revert a completed item to open state.

undoAction : it undoes the late committed action on the item

redoAction : it allows users to redo the most recent undo action on the item

undoListAction : This allows users to undo the most recent action done on the list

redoListAction : This allows users to redo the most recent action done on the list.

# Friend Manager:

sendFriendRequest : This allows users to send friend requests to users of the toDoManger app. On sending request, user will be sharing his/her details.

CheckRequest : This allows the users to see a list of pending/new friend requests they received

acceptFriend : This allows users to accept friends from pending/new list of requests. Adding a user as a friend gives them authorization to create, read, edit and delete items and read only for lists.

# Socket.io

**Emit :**

* verifyUser : emitted by the socket to get authentication details of the user that just logged in.
* error-occurred : on error the socket emits this with error information
* online-user-list : on login and logout socket emits the updated online users list to the room
* userId : on user Id the socket emits the following data on various requests from the client
  + for ‘get-online-users’ event from client : socket emits online users list on the userId requesting the data
  + for ‘get-user-list’ event from client : socket emits user’s lists on the userId requesting the data
  + for ‘get-list-item’ event from client : socket emits list’s items on the userId requesting the data
  + for ‘get-friends’ event from client : socket emits user’s friends data on userId requesting the data

**Listening :**

* auth-user : on getting auth-user from client it sets the new user online and updates the online user list.
* get-online-users : on getting this event it sends back the online user list to client on the userId of the requester.
* get-user-lists : on getting this event from the client it sends back the user’s lists data on the userId of the requester.
* get-list-items : on getting this event from the client it sends back the items of the list on the userId of the requester.
* get-friends : on getting this event from the client it sends back the user’s friends details to the userId of the requester.
* Join-friends-rooms : on getting this event from the client the socket searches for all the user’s friends details and joins their rooms
* Friend-request : on getting this event from the client the socket searches for the users details and composes the notification details and broadcasts it on the user’s roomId, so that all the user’s friends listening to this room Id will get the notification.
* List-notification : on getting this event from the client the socket composes the notification of the action performed on the list and its content and broadcasts it to all the user that are friends of the user.
* Disconnect : on getting this event from the client, the socket removes the user from the onlineUsers list and broadcasts the updated onlineUsers list to the room “todoUsers”.
  + It also removes the users from their friends rooms.