# 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:

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
* error-occurred
* online-user-list : on login and logout
* get-user-friends (eventEmitter)
* get-user-list : get be used on login or when a user selects a friend to get lists
* get-list (eventEmitter)
* userId: (myId.emit): to emit userlists on login
* userId : (myId.emit) : to emit userFriends details on selecting user
* friend-request-notification : broadcasted to socket.room

**Listening :**

* auth-user
* get-user-lists
* sent-friend-request : after a success response on sending a friend request this event is emitted by the client for creating notification.

On Login :

1. On login socket sends “**verifyUser”**
2. As response client sends “**auth-user”** with authToken
3. To that socket checks the authToken and adds the user to online list
   1. Socket broadcasts “**onlineUsers”** to the room “*toDoUser***”**
   2. it emits an event on eventEmitter “*get-user-friends*” which gets the friends of the current user
   3. it sends the **users friends** details on “userId”
4. Client sends  **“get-user-lists”** with listRequest object containing userId and ownerId.
5. As a response socket sends userList on **“userId”** of the sender

Get-lists :

Client emits this event after verifying authToken.