\*\*Here will be two types of users:

1. User
2. Member

—----------------------credentials—------------------------

* **~~[POST/users/signup]~~** ~~A new user can signup and should return a token with email and role property and memberCopID.~~
* ~~If a new user has a memberCopID, make the user member by adding this email to the existing member.(based on memberCopID).~~
* **~~Old email of the member should be saved.~~**
* **~~[POST/users/login]~~** ~~An existing user could log in and get a token.~~
* **~~[GET/users/me]~~** ~~will get all information about the user.(get user using token)~~
* **~~[GET/users]~~** ~~will get all users. Must verify authorization.[sort,filter]~~
* **~~[GET/users/:id]~~** ~~get user by id. Must verify authorization.~~
* **~~[PUT/users/active/:id]~~** ~~activate the user by id and deactivate the previous activated user. Must verify authorization.~~
* **~~[DELETE/users/:id]~~** ~~inactive the user by id. Must verify authorization.~~

—-------------------------Members------------------------------------

* **~~[POST/members/addnew]~~** ~~An admin or other authorised member could add a new member. Where memberCopID,name,status mandatory and email,mobile,fatherName,motherName, address,bloodGroup,accountOpening is optional.~~
* **~~[PATCH/members/:id]~~** ~~will update member information. Must verify token and user authorization [only member itself (if member is active and has admin approval),admin and others authorised can update member]. Only authorised members could delete.~~
* **~~[DELETE/members/:id]~~** ~~will delete a member. Only authorised~~
* **~~[GET/members]~~** ~~get all members. Must verify authentication. [sort,filter,limit,skip]~~
* **~~[GET/member/:id]~~** ~~get specific member information.~~

—----------------------Committee - Fully authorisation verified---------------------------------

* **~~[POST/committee]~~** ~~admin could add new committees. This should contain committeeElectedOn,status,committees [ where committee members info] and expiredOn [it will be based on input].~~
* ~~Every committee member should contain committee info in their member collection like- committee:[{status,role,electionTime}].~~
* ~~When new committees are added, existing committees should be in expired status.~~
* **[PATCH/committee/addmember]** will update the active committee and add a new member on it where input must be {name,role[active,inactive,paused,removed],status,memberCopID}.
* **[PATCH/committee]** will update information on the active committee.
* **[PATCH/committee/:id]** update a specific committee.
* **[Delete/committee/committeeID]** will delete a committee.
* **[GET/committee]** get active committee only.
* **[GET/committee/all]** get all committees.[sort,filter]
* **[GET/committee/:id]** get a specific committee.

—-------------------------Only admin authorised-----------------------------------

* **[POST/admin/:id]** make a member admin.
* **[GET/admin]** get all admin [sort,filter,skip,limit]
* **[DELETE/admin/:id]** delete an admin

—-------------------------Deposits---------------------------

Status: [reject,approved,pending]

* **[POST/finance/deposit/add]** add new deposit {memberCopID,depositAmount,status,collector{collectorName,collectionTime,memberCopID},dataEntry{dataEntrierName, entryTime, memberCopID}, authorised{authoriserName, authorisationTime,memberCopID}}, if data entried by collector or data entrier or others except chairman, md, manager that should go to approval request.
* **[GET/finance/deposit/request]** get all unapproved deposit requests.(Only authorised) [sort,filter,skip,limit]
* **[POST/finance/deposit/approve/:id]** authorised roles could approve a deposit request at this route.
* **[POST/finance/deposit/reject/:id]** reject a deposit request.
* **[GET/finance/deposit/:id]** get specific deposit (authorised and member itself)
* **[GET/finance/deposits]** get all deposits (authorised only).[sort,filter,skip,limit]
* **[GET/finance/deposits/member/:id]** get all deposits of a member (authorised and member itself).deposit of a member should contain total deposit amount also.
* **[DELETE/finance/deposit/:id]** delete a deposit (authorised only).
* It should be confirmed individual deposit has been deleted

—-------------------------withdraw---------------------------

Status: [reject,approved,pending]

* **[POST/finance/withdraw/add]** add new withdraw {memberCopID,withdrawAmount,status,witness{witnessName,time,memberCopID},dataEntry{dataEntrierName, entryTime, memberCopID}, authorised{ authorisedName, authorisationTime,memberCopID}}, if data entered by collector or data entries or others except chairman, md, manager that should go to approval request.
* **[GET/finance/withdraw/request]** get all unapproved withdrawal requests.(Only authorised) [sort,filter,skip,limit]
* **[POST/finance/withdraw/approve/:id]** authorised roles could approve a withdrawal request at this route.
* **[POST/finance/withdraw/reject/:id]** reject a withdrawal request.
* **[GET/finance/withdraw/:id]** get specific withdrawal(authorised and member itself)
* **[GET/finance/withdraw]** get all withdrawal(authorised only).[sort,filter,skip,limit]
* **[GET/finance/withdraw/member/:id]** get all withdrawals of a member (authorised and member itself).withdraw of a member should contain total withdraw amount also.
* **[DELETE/finance/withdraw/:id]** delete a withdraw (authorised only).
* It should be confirmed individual withdrawal has been deleted

—-------------------------investment---------------------------

Status: [reject,approved,pending]

* **[POST/finance/investment/add]** add new investment {investmentAmount{initialInvest,additionalInvest[amount,purpose,date],individualInvest[{memberCopID,amount}]},status,investmentDate,platform{platformName,nameOfManager,contactNumber},dataEntry{dataEntrierName, entryTime, memberCopID}, authorised{ authoriserName, authorisationTime,memberCopID}}, if data entered by collector or data entries or others except chairman, md, manager that should go to approval request.
* When a new investment is added, add this in individual members.
* **[POST/finance/investment/add/additional]** add an additional investment based on investment id(Push to additionalInvest array of investment).
* **[GET/finance/investment/request]** get all unapproved investment requests.(Only authorised) [sort,filter,skip,limit]
* **[POST/finance/investment/approve/:id]** authorised roles could approve an investment request at this route.
* **[POST/finance/investment/reject/:id]** reject an investment request.
* **[GET/finance/investment/:id]** get specific investment(authorised and member itself)
* **[GET/finance/investment]** get all investment(authorised only).[sort,filter,skip,limit]
* **[GET/finance/investment/member/:memberCopID]** get all investment of a member (authorised and member itself). Investment of a member should contain a total investment amount also.
* **[DELETE/finance/investment/:id]** delete an investment (authorised only).
* It should be confirmed individual investment has been deleted

—-------------------------Profit---------------------------

Status: [reject,approved,pending]

* **[POST/finance/profit/add]** add new profit. Profit must be obtained from an investment. {investmentID,investmentAmount,closingTotalAmount,profitAmount,individualProfit[{memberCopID,profitAmount}],dataEntry{dataEntrierName, entryTime, memberCopID}, authorised{ authoriserName, authorisationTime,memberCopID}}
* When a new profit is added, add this in individual members.
* **[GET/finance/profit/request]** get all unapproved profit requests.(Only authorised) [sort,filter,skip,limit]
* **[POST/finance/profit/approve/:id]** authorised roles could approve a profit request at this route.
* **[POST/finance/profit/reject/:id]** reject a profit request.
* **[GET/finance/profit/:id]** get specific profit(authorised and member itself)
* **[GET/finance/profit]** get all profit(authorised only).[sort,filter,skip,limit]
* **[GET/finance/profit/member/:memberCopID]** get all profit of a member (authorised and member itself).profit of a member should contain total profited amount also.
* **[DELETE/finance/profit/:id]** delete an investment (authorised only).
* It should be confirmed individual profit has been deleted.

—-------------------------Expenses---------------------------

Status: [reject,approved,pending]

* **[POST/finance/expense/add]** add new expense. Expense must be obtained from an investment (if investment id provided).
* When a new expense is added, add this in individual members.
* **[GET/finance/expense/request]** get all unapproved expense requests.(Only authorised) [sort,filter,skip,limit]
* **[POST/finance/expense/approve/:id]** authorised roles could approve an expense request at this route.
* **[POST/finance/expense/reject/:id]** reject an expense request.
* **[GET/finance/expense/:id]** get specific expense(authorised and member itself)
* **[GET/finance/expense]** get all expenses(authorised only).[sort,filter,skip,limit]
* **[GET/finance/expense/member/:memberCopID]** get all expenses of a member (authorised and member itself).expense of a member should contain total expensed amount also.
* **[DELETE/finance/expense/:id]** delete an expense(authorised only).
* It should be confirmed that individual expenses have been deleted.

—-------------------------account---------------------------

* **[GET/account/:memberCopID]** get all about finance. All deposit,invest,profit,withdraw with total calculation.

| User | |
| --- | --- |
| name | String |
| email | String |
| password | String |
| photoURL | String |

| Member | |
| --- | --- |
| memberCopID | String |
| name | String |
| email | Object  {  Default,old[]  } |
| fatherName | String |
| motherName | String |
| mobile | String |
| address | String |
| deposits | Array  [{  depositAmount, date,  moreAboutDeposit  }] |
| withdrawals | Array  [{  withdrawAmount, date,  moreAboutWithdraw  }] |
| investments | Array  [{  investedAmount, date,  status,  moreAboutInvest  }] |
| profits | Array  [{  profitedAmount, date, moreAboutInvestment,  moreAboutProfit  }] |
| expense | Array  [{  depositAmount,  reason, date,  moreAboutExpense  }] |

| Deposit | |
| --- | --- |
| memberCopID | String |
| name | String |
| moreAboutMember | ObjectId |
| depositAmount | Number |
| status | String |
| collector | Object  {  name, date, memberCopID,  moreAboutCollector  } |
| dataEntry | Object  {  dataEntrierName, entryTime, memberCopID,  moreAboutDataEntrier  } |
| authorised | Object  {  authoriserName, authorisationTime, memberCopID,  moreAboutAuthoriser  } |

| Withdraw | |
| --- | --- |
| memberCopID | String |
| name | String |
| moreAboutMember | String |
| status | String |
| witness | Object  {  collectorName, collectionTime, memberCopID,  moreAboutWitness  } |
| dataEntry | Object  {  dataEntrierName, entryTime, memberCopID,  moreAboutDataEntrier  } |
| authorised | Object  {  authoriserName, authorisationTime, memberCopID,  moreAboutAuthoriser  } |

| Investment | |
| --- | --- |
| investedAmount | Object  {initialInvest,  additionalInvest [{amount, date, purpose}]} |
| status | String |
| individualInvest | Array  [{name, memberCopID, amount}] |
| investmentDate | String |
| platform | Object  {  platformName, nameOfManager, contactNumber  } |
| dataEntry | Object  {  dataEntrierName, entryTime, memberCopID, moreAboutDataEntrier  } |
| authorised | Object  {  authoriserName, authorisationTime, memberCopID,  moreAboutAuthoriser  } |
| profits | Object  {  profitID, amount ,date, moreAboutProfit  } |
| expenses | Array  [{amount,reason,date, moreAboutExpense}] |

| Profit | |
| --- | --- |
| collectedTotalAmount | Number |
| totalInvest | Object |
| totalExpense |  |
| profit | Number |
| status | String |
| individualProfit | ArrayOfObject  [{memberCopID, profitAmount}] |
| investmentDate | String |
| dataEntry | Object  {  dataEntrierName, entryTime, memberCopID  } |
| authorised | Object  {  authoriserName, authorisationTime, memberCopID  } |

| Expenses | |
| --- | --- |
| expensedAmount | String |
| purpose | String |
| status | String |
| individualExpense | ArrayOfObject  [{memberCopID, expensedAmount}] |
| expenseDate | String |
| dataEntry | Object  {  dataEntrierName, entryTime, memberCopID  } |
| authorised | Object  {  authoriserName, authorisationTime, memberCopID  } |
| investment | Object  {  totalInvestmentID,  } |