# COCO - Professional Anonymous Network
(Last updated on 17th May 2020 by Nitin Mishra)

## FUNCTIONAL REQUIREMENTS:

1. User can enter mail (preferable work mail) and can verify same using received OTP on mail
2. Once verified, user gets read-only or full access based on his mail domain type and company
3. User can choose password & available username or system will assign username randomly
4. User will have to choose a country and enter his department, designation to create profile
5. Platform will not store any user information so that nobody could trace any user
6. Updating a username should also update profile link but not referral link (linked with user_id)
7. User should be able to follow and unfollow companies and channels
8. User can see all/followed companies, all/followed channels, bookmarked posts, blocked user
9. User starts following his own company automatically after successful registration
10. User can create/tag/upvote/downvote/delete/bookmark/unbookmark/share a post
11. User can comment on post, reply on comment and upvote/downvote/delete/share comment
12. User can either upvote/downvote a post or a comment at most once
13. User should be able to give bounty to any post creator or comment creator
14. Users creating a post or a comment will start following that post & channel automatically
15. A post will get mapped to channel tags also apart from post tags & company tags (if any)
16. Each post/comment will show last update time (7d/9h/3m), username, up/down vote counts
17. Sharing a post or comment on whatsapp should create screenshot with referral join deeplink
18. User can search relevant channels companies and posts by keyword
19. Bounties can be earned on create post, comment/upvote/downvote & successful referrals
20. Bounties are transferable across the platform to users
21. Rank of users/companies/channels/tags should also be calculated for analytics purpose
22. User can see its rank and other analytics metrics on profile page
23. User should be able to flag/unflag a post/comment and block/unblock all posts from a user
24. If abuse count for comment/post reaches 3/10, show "this comment/post has been deleted
25. User can subscribe for job notifications @ ~~INR 499~~ INR 99 per year per job role
26. User can send direct message to other user or message in a group chat (Phase 2)
27. User can see median salaries across other companies for same designation (Phase 3)
28. User can put a bid and pay to other users who help him to crack interview rounds (Phase 4)

## SYSTEM REQUIREMENTS:

Highly available system (ensuring no single point of failure)
Low latency APIs (<100ms)
Analytics data

## DISTRIBUTION STRATEGY:

Corporate employees anonymously inviting their colleagues on the platform (word of mouth)

## USER ACQUISITION:

1K users in a week, 10K users in a month, 1Lac in 3 Months and 1M in 6 Months

**DB SCHEMA (POSTGRES):**

**tbl_access ->** access_id(pk), companyid(fk), domain(index), access_type <readonly/full>, last_updated_epoch

**tbl_user ->** user_id(pk), username(index), workmail_hash(index), last_otp_hash(index), last_otp_expire_epoch, designation, companyid(fk), depid(fk), profile_link, referred_by_link, referral_link(unique), user_rank_id(fk), pass_hash(index), is_mail_verified <0/1>, access_id(fk), issignedin <0/1>, successful_referral_count, bounties_received_count, bounties_consumed_count, bounties_left_count, posts_create_count, comment_gave_count, comment_received_count, users_i_reported_count, users_reported_me_count, posts_i_reported_count, my_posts_got_reported_count, comments_i_reported_count, my_comments_got_reported_count, upvote_gave_count, upvote_received_count, downvote_gave_count, downvote_received_count, user_last_activity_date, username_updated_epoch, last_updated_epoch

**tbl_rank ->** user_rank_id(pk), user_rank<bronze/silver/gold/platinum/diamond>, min_bounty <10/100/1000/10000/100000>, max_bounty <99/999/9999/99999/999999>

**tbl_country ->** countryid(pk), country_name, last_updated_epoch

**tbl_department ->** depid(pk), dep_name, last_updated_epoch

**tbl_company ->** companyid(pk), countryid(fk), company_name, tagid(fk), company_rank, company_user_count, last_updated_epoch

**tbl_dep_company_mapping ->** companyid(pf), depid(pf), last_updated_epoch *(N:N mapping)*

**tbl_followed_company_mapping ->** user_id(pf), companyid(pf), last_updated_epoch *(N:N mapping)*

**tbl_channel_admin ->** channelid(pk), channel_name, channel_rank, channel_post_count, last_updated_epoch

**tbl_followed_channel_mapping ->** user_id(pf), channelid(pf), last_updated_epoch *(N:N mapping)*

**tbl_post ->** postid(pk), channelid(fk), posterid(fk), post_title, post_data, isabusivepost <0/1>, post_comment_count, post_bounty_count, post_upvote_count, post_downvote_count, post_link, post_share_count, post_abuse_count, last_updated_epoch

**tbl_voted_post_mapping ->** user_id(pf), postid(pf), vote_post_type<up/down>, last_updated_epoch *(N:N mapping)*

**tbl_voted_comment_mapping ->** user_id(pf), commentid(pf), vote_comment_type<up/down>, last_updated_epoch *(N:N mapping)*

**tbl_followed_post_mapping ->** user_id(pf), postid(pf), last_updated_epoch *(N:N mapping)*

**tbl_bookmarked_post_mapping ->** user_id(pf), postid(pf), last_updated_epoch *(N:N mapping)*

**tbl_comment ->** comment_id(pk), comment_data, comment_typeid(fk), postid(fk), comment_abuse_count, isabusivecomment <0/1>, prev_commentid(fk), commenterid(fk), comment_bounty_count, comment_upvote_count, comment_downvote_count, comment_share_count, comment_link, last_updated_epoch

**tbl_comment_type ->** comment_typeid(pk), comment_type<new/reply/moderator>, last_updated_epoch

**tbl_hashtag ->** tagid(pk), tag_name, tag_typeid(fk), tag_rank, tag_used_count, last_updated_epoch

**tbl_hashtag_type ->** tag_typeid(pk), tag_type <company/channel/post>, last_updated_epoch

**tbl_tagged_channel_mapping ->** channelid(pf), tagid(pf), last_updated_epoch *(N:N mapping)*

**tbl_tagged_post_mapping ->** postid(pf), tagid(pf), last_updated_epoch *(N:N mapping)*

**tbl_blocked_user_mapping->** reporterid(pf), reporteeid(pf),last_updated_epoch *(N:N mapping)*

**tbl_reported_post_mapping ->** postid(pf), reporterid(pf), last_updated_epoch *(N:N mapping)*

**tbl_reported_comment_mapping ->** commentid(pf), reporterid(pf), last_updated_epoch *(N:N mapping)*

## SERVER SIDE CACHE (REDIS):

1. Faster User Level Analytics: User analytics data for all users
2. Faster Company Level Analytics: Company analytics data for all companies
3. Faster Tag Level Analytics: Tag analytics data for all tags
4. Faster Post Search: List of all posts and its data by keyword search
5. Faster Job Search: List of all vacancies and its data by role search
6. Faster Login: Workmail_hash & pass_hash for all users

## CLIENT SIDE CACHE (FLUTTER):

1. List of channels and List of companies
2. All my bounties, username, rank, company, department, designation and workmail_hash
3. List of posts and its data along with my up/down votes for *trending* and *for-you* channels
4. List of posts and its data along with my up/down votes in bookmarked section

**BACKEND APIS (GOLANG):**

**POST /v1/user/getotp/<countryid>/<workmail>/<referred_by_link> :** create user_id, workmail_hash, otp_hash

**POST /v1/user/verifyotp/<otp> :** if mail is verified: assign proper access_id to user_id

**GET /v1/user/isusernameavailable/<user_id>/<username> :** if username does not exist or if username is already assigned to his user_id, username is considered as available to user_id

**POST /v1/user/signup/<workmail>/<pass>/<username> :** creates pass_hash for user_id

**POST /v1/user/signin/<workmail>/<pass> :** matches workmail_hash & pass_hash

**POST /v1/user/signout/<user_id> :** updates issignedin from 1 to 0

**POST /v1/user/invite/<user_id>/<workmail> :** sends anonymous invite on mail ("India's most trusted anonymous professional network" mailer having "get OTP" button calling getotp API)

**POST /v1/user/update/<userid>/<username>/<designation>/<depid> :** update profile link also whenever username gets updated in user table

**POST /v1/user/block/<reporterid>/<reportee_id> :** create in blocked_user_mapping

**POST /v1/user/unblock/<reporterid>/<reportee_id> :** del from blocked_user_mapping

**GET /v1/user/blocked/<user_id> :** get list of users blocked by user_id

**GET /v1/user/stats/<user_id> :** get users profile stats by user_id

**GET /v1/user/refer/<user_id> :** get unique referral link

**GET /v1/user/referred/<user_id> :** get list of successful referrals by userid

**POST /v1/company/follow/<user_id>/<companyid> :** create in followed_company_mapping

**POST /v1/company/unfollow/<user_id>/<companyid>:** del from followed_company_mapping

**GET /v1/company/followed/<user_id> :** get list of companies followed by user_id

**GET /v1/company/list/<countryid> :** fetch list of all companies for a country

**POST /v1/channel/follow/<user_id>/<channelid> :** add record in followed_channel_mapping

**POST** **/v1/channel/unfollow/<user_id>/<channelid>** : followed_channel_mapping

**GET** **/v1/channel/followed/<user_id>** : get list of channels followed by user_id

**GET** **/v1/channel/list/** : fetch list of all channels

**POST** **/v1/hashtag/create/<user_id>/<tag_name>** : create a tag on post based on access_id

**GET** **/v1/hashtag/search/<keyword>** : fetch matching list of tag_name by keyword

**GET** **/v1/post/search/<tag_name>** : fetch list of all post_id by tag_name

**POST** **/v1/post/follow/<user_id>/<postid>** : add record in followed_post_mapping table

**POST** **/v1/post/unfollow/<user_id>/<postid>** : delete record from followed_post_mapping

**GET** **/v1/post/followed/<user_id>** : fetches list all posts followed

**POST** **/v1/post/create/<poster_id>/<channel_id>/<post_title>/<post_data>** : updates user, channel, post and followed_post_mapping tables based on access_id

**POST** **/v1/post/delete/<post_id>** : updates user, channel, post and followed_post_mapping tables based on access_id (user confirmation required on UI)

**POST** **/v1/post/tag/<user_id>/<post_id>/<tag_id>** : updates tagged_post_mapping table

**POST** **/v1/post/vote/<user_id>/<post_id>/<vote_type>** : updates user, post and voted_post_mapping (delete record if vote_type is null) based on access_id

**POST** **/v1/post/bounty/<user_id>/<post_id>** : updates user and post tables, transacts bounty

**POST** **/v1/post/abuse/<user_id>/<post_id>** : updates reported_post_mapping, user & post

**POST** **/v1/post/unabuse/<user_id>/<post_id>** : updates reported_post_mapping, user & post

**GET** **/v1/post/abused/<user_id>** : fetches all abused posts

**POST** **/v1/post/bookmark/<user_id>/<post_id>** : create in bookmarked_post_mapping table

**POST** **/v1/post/unbookmark/<user_id>/<post_id>** : del from bookmarked_post_mapping

**GET** **/v1/post/bookmarked/<user_id>** : get list of all bookmarked post_id

**GET** **/v1/post/fetch_by_post/<post_id>** : fetch post data by post_id

**GET   /v1/post/fetch_by_channel/<channel_id> :** fetch list of all post_id for a channel

**GET   /v1/post/share/<post_id> :** get shareable post_link

**GET   /v1/post/isvoted/<user_id>/<post_id> :** check if user has voted on post

**POST /v1/comment/create/<commenterid>/<postid>/<prev_commentid>/<comment_data>
:** updates user, comment, post and followed_post_mapping tables based on access_id

**POST /v1/comment/delete/<comment_id> :** updates user, comment, post and
followed_post_mapping tables (user confirmation required on UI)

**POST /v1/comment/vote/<user_id>/<comment_id>/<vote_type> :** updates user, post,
comment & voted_comment_mapping (delete record if vote_type is null) based on access_id

**POST /v1/comment/bounty/<user_id>/<comment_id> :**  updates user and comment tables

**POST /v1/comment/abuse/<user_id>/<comment_id>
:** updates reported_comment_mapping, user and comment tables

**POST /v1/comment/unabuse/<user_id>/<comment_id>
:** updates reported_comment_mapping, user and comment tables

**GET   /v1/comment/abused/<user_id> :** fetches all abused comments

**GET   /v1/comment/isvoted/<user_id>/<comment_id> :** check if user has voted on comment

**GET   /v1/comment/share/<comment_id> :** get shareable comment_link