COCO - India’s Most Trusted Anonymous Professional Network

(Last updated on 18th May by Nitin Mishra)

FUNCTIONAL REQUIREMENTS (Phase 1):

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 will update profile link but not long referral link (link to mail_hash)
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 (6d/9h/3m), username, up/down vote counts
17. Sharing a post or comment on whatsapp should create screenshot with long referral join link
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 3/6, show “this comment/post has been deleted
25. User can become pro user by paying @ ~~INR 499~~ INR 99 per year
26. Pro users can opt to receive job notifications for the roles they are interested in
27. Pro users can send direct messages to other users

Phase 2 : Pro users can see company salary graphs for all roles (post sharing his role & salary)
Phase 3 : User can put a bid and pay to other users who help him to crack interview rounds

SYSTEM REQUIREMENTS:

Highly available system (ensuring no single point of failure)
Low latency APIs (<=99ms)
Analytics data

DISTRIBUTION STRATEGY:

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

TARGET MILESTONES:

900 users in a week, 9K users in a month, 9Lac in half year and 9M in a year (post launch)

DB SCHEMA (POSTGRES):

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

tbl_user -> mail_hash(pk), username(index) , last_otp_hash(index), last_otp_expire_epoch, designation, company_id(fk), dep_id(fk), profile_link, long_referred_by_link, long_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>, next_milestone_bounty <99/999/9999/99999/999999>

tbl_country -> country_id(pk), country_name, last_updated_epoch

tbl_department -> dep_id(pk), dep_name, last_updated_epoch

tbl_company -> company_id(pk), country_id(fk), company_name, tag_id(fk), company_rank, company_user_count, last_updated_epoch

tbl_dep_company_mapping -> company_id(pf), dep_id(pf), last_updated_epoch *(N:N)*

tbl_followed_company_mapping -> mail_hash(pf), company_id(pf), last_updated_epoch *(N:N mapping)*

tbl_channel_admin -> channel_id(pk), channel_name, channel_rank, channel_post_count, last_updated_epoch

tbl_followed_channel_mapping -> mail_hash(pf), channel_id(pf), last_updated_epoch *(N:N mapping)*

tbl_post -> post_id(pk), channel_id(fk), poster_mail_hash(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 -> mail_hash(pf), post_id(pf), vote_post_type<up/down>, last_updated_epoch *(N:N mapping)*

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

tbl_followed_post_mapping -> mail_hash(pf), post_id(pf), last_updated_epoch *(N:N)*

tbl_bookmarked_post_mapping -> mail_hash(pf), post_id(pf), last_updated_epoch *(N:N)*

tbl_comment -> comment_id(pk), comment_data, comment_type_id(fk), post_id(fk), comment_abuse_count, isabusivecomment <0/1>, prev_comment_id(fk), commenter_mail_hash(fk), comment_bounty_count, comment_upvote_count, comment_downvote_count, comment_share_count, comment_link, last_updated_epoch

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

tbl_hashtag -> tag_id(pk), tag_name, tag_type_id(fk), tag_rank, tag_used_count, last_updated_epoch

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

tbl_tagged_channel_mapping -> channel_id(pf), tag_id(pf), last_updated_epoch *(N:N mapping)*

tbl_tagged_post_mapping -> post_id(pf), tag_id(pf), last_updated_epoch *(N:N mapping)*

tbl_blocked_user_mapping-> reporter_mail_hash(pf), reportee_mail_hash(pf), last_updated_epoch *(N:N mapping)*

tbl_reported_post_mapping -> post_id(pf), reporter_mail_hash(pf), last_updated_epoch *(N:N mapping)*

tbl_reported_comment_mapping -> comment_id(pf), reporter_mail_hash(pf), last_updated_epoch *(N:N)*

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: mail_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 mail_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/<country_id>/<mail>/<long_referred_by_link> : create mail_hash, otp_hash

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

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

POST /v1/user/signup/<mail>/<pass>/<username> : creates pass_hash for mail_hash

POST /v1/user/signin/<mail>/<pass> : matches mail_hash & pass_hash

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

POST /v1/user/invite/<mail_hash>/ : users can share long referral link with colleagues on whatsapp which opens invite webpage asking referee to *enter work mail* and click on *get OTP*)

POST /v1/user/update/<mail_hash>/<username>/<designation>/<dep_id> : update profile link also whenever username gets updated in user table

POST /v1/user/block/<reporter_mail_hash>/<reportee_mail_hash> : create in blocked_user_mapping

POST /v1/user/unblock/<reporter_mail_hash>/<reportee_mail_hash> : del from blocked_user_mapping

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

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

GET /v1/user/refer/<mail_hash> : get unique long referral link

GET /v1/user/referred/<mail_hash> : get list of successful long referrals by mail_hash

POST /v1/company/follow/<mail_hash>/<company_id> : create in followed_company_mapping

POST /v1/company/unfollow/<mail_hash>/<company_id>: del from followed_company_mapping

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

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

POST /v1/channel/follow/<mail_hash>/<channel_id> : add record in followed_channel_mapping

POST /v1/channel/unfollow/<mail_hash>/<channel_id> : followed_channel_mapping

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

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

POST /v1/hashtag/create/<mail_hash>/<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/<mail_hash>/<post_id> : add record in followed_post_mapping table

POST /v1/post/unfollow/<mail_hash>/<post_id> : delete record from followed_post_mapping

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

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

POST /v1/post/delete/<mail_hash><post_id> : updates user, channel, post and followed_post_mapping tables based on access_id (delete confirmation required on UI)
NOTE: mail_hash should be equal to poster_mail_hash (user can delete his own post only)

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

POST /v1/post/vote/<mail_hash>/<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/givebounty/<mail_hash>/<post_id> : updates user and post tables

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

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

GET /v1/post/abused/<reporter_mail_hash> : fetches all abused posts

POST /v1/post/bookmark/<mail_hash>/<post_id> : create in bookmarked_post_mapping

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

GET /v1/post/bookmarked/<mail_hash> : 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/<mail_hash>/<post_id> : check if user has up/down voted on post

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

POST /v1/comment/delete/<mail_hash>/<comment_id> : updates user, comment, post and followed_post_mapping tables (delete confirmation required on UI)
NOTE: mail_hash should be equal to commenter_mail_hash (can delete his own comment only)

POST /v1/comment/vote/<mail_hash>/<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/givebounty/<mail_hash>/<comment_id> : updates user and comment tables

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

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

GET /v1/comment/abused/<reporter_mail_hash> : fetches all abused comments

GET /v1/comment/isvoted/<mail_hash>/<comment_id> : check if user has up/down voted

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