COCO - Professional Anonymous network

(Last updated on 17th May 2020 by Nitin Mishra)

Functional requirements (PRD):

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 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. Changing a username will change profile link but not referral link (being linked with user_id)
7. User can see all companies and channels availables on the platforms
8. User should be able to follow and unfollow companies and channels
9. User starts following his own company automatically after successful registration
10. User should be able to create a post in a channel, tag a post, upvote/downvote/share post
11. User can comment on post, reply on comment and upvote/downvote/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/channel/company can have multiple tags attached to it
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 any post or comment and can also block all posts from a user
24. If abuse count for comment reaches 3, show it as “this comment has been deleted by mod”
25. If abuse count for post reaches 10, show it as “this post has been taken down”
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)

Performance Requirements:

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

Distribution strategy:

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

Target:

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

DB schema based on Entity relation (postgres):

tbl_auth-> user_id(pk), countryid(fk), mail_hash(index), last_otp_hash(index), last_otp_expire_epoch, pass_hash(index), is_mail_verified<0/1>, access_id(fk), canvote<0/1>, canpost<0/1>, cancomment<0/1>, isloggedin<0/1>, last_updated_epoch

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

tbl_user-> hash(pk), user_id(fk), username(index), designation, companyid(fk), deptid(fk), profile_link, referral_link, user_rank_id(fk), 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-> deptid(pk), dept_name, last_updated_epoch

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

tbl_dept_company_mapping-> companyid(pf), deptid(pf), last_updated_epoch *(N:N mapping)*

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

tbl_channel-> 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_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), reportee_id(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)*

APIs (golang apis with JWT auth token):

POST /v1/auth/getOTP/<countryid>/<workmail>
POST /v1/auth/verifyOTP/<user_id>/<OTP> => check last-otp-hash before expiry
POST /v1/auth/login/<workmail>/<pass> => updates isloggedin to 1
POST /v1/auth/logout/<user_id> => updates isloggedin to 0

GET   /v1/user/isusernameavailable/<username> =>checks if username available
POST /v1/user/create/<user_id>/<pass>/<username>/<designation>/<companyid>/<deptid>
POST /v1/user/update/<user_id>/<username> => update username and profile link in user table
POST /v1/user/block/<reporterid>/<reportee_id> => updates blocked_user_mapping
GET   /v1/user/refer/<user_id>
POST /v1/company/follow/<user_id>/<companyid>=>add record in followed_company_mapping
POST /v1/company/unfollow/<user_id>/<companyid>
=> delete record from followed_company_mapping
GET   /v1/company/list/<countryid>
POST /v1/channel/follow/<user_id>/<channelid> => add record in followed_channel_mapping
POST /v1/channel/unfollow/<user_id>/<channelid>
=> delete record from followed_channel_mapping
GET   /v1/channel/list/
POST /v1/hashtag/create/<tag_name> => creates hashtag with tag_type as post
GET   /v1/hashtag/search/<tag_name>/<tag_typeid>
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 table
POST /v1/post/create/<poster_id>/<channel_id>/<post_data>
=> updates user, channel, post and followed_post_mapping tables
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)
POST /v1/post/bounty/<user_id>/<post_id> => updates user and post tables
POST /v1/post/abuse/<user_id>/<post_id> => updates reported_post_mapping, user & post
POST /v1/post/bookmark/<user_id>/<post_id> => updates bookmarked_post_mapping table
GET   /v1/post/fetch_by_post/<post_id>
GET   /v1/post/fetch_by_channel/<channel_id>
GET   /v1/post/share/<post_id>
GET   /v1/post/isvoted/<user_id>/<post_id>
POST /v1/comment/create/<commenter_id>/<post_id>/<prev_comment_id>/<comment_data>
=> updates user, comment, post and followed_post_mapping tables
POST /v1/comment/vote/<user_id>/<comment_id>/<vote_type>
=> updates user, post, comment & voted_comment_mapping (delete record if vote_type is null)
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
GET   /v1/comment/isvoted/<user_id>/<comment_id>
GET   /v1/comment/share/<post_id>/<comment_id>

24 hour cached API response (memcache):

GET   /v1/post/trending/<country_id>
GET   /v1/mailer/personalized/<user_id>