MyInstagram Query List

This document consists of all the queries essential to perform the use cases mentioned above. Queries for each use case are listed below.

- Sign Up:
  a. Find if email/phone no/username is already present in user table or not
  b. If not already present → Create new entry in user table and insert email/phone no, username, password, DOB, fullname, createdAt

- Login:
  a. Find username/email/phone no in user table
  b. If present, compare password with username's corresponding password
  c. if(account==disable) for u_id, execute query-26
  d. Return user

- Update Profile:
  a. If updated Email, Phone no. or username, find if already present in User
  b. If not already present, update email, phone no. or username accordingly for u_id
  c. If updated other attributes than the above mentioned, update for u_id
  d. Return updated values

- Post (Images/Videos):
  a. Create new entry in post table and insert u_id, content [ ], title, caption, location, tagged[u_id : type], type(Reel/Post/IGTV), createdAt, modifiedAt, hashTags [ ]
  b. Find u_id in user table → Increment postsCount and insert p_id in posts [ ]
  c. For hashTags [ ] ⇒ Find hashTag in HashTag Table → if(present) ? insert p_id in posts [ ] : create new entry in HashTag and insert hashtag, p_id
  d. When trying to tag people, execute this query (user1 tags user2): find user2_id in User Table then return allowTagsFrom

  Return - Updated values

- Post(Reel):
  a. Create new entry in post table and insert u_id, createdAt, modifiedAt, caption, location, tagged[u_id : type], type(Reel/Post/IGTV), hashTags[ ], content [ ],

music
b. Find u_id in user table → Increment postsCount and insert p_id in posts [ ]
c. When trying to tag people, execute this query (user1 tags user2): find user2_id in User Table then return allowTagsFrom
d. For hashTags [ ] ⇒ Find hashTag in HashTag Table → if(present) ? insert p_id in posts [ ] : create new entry in HashTag and insert hashtag, p_id

Return - Updated values

- Like/Unlike(Post/Comment):
  a. Find p_id/comment_id and update likes and likedBy[u_id]
  b. if(Post) find u_id in User Table then insert/remove p_id in/from liked[ ]

  Return - Updated values

- Comment:
  a. Create new entry in comment table and insert u_id, p_id, owner_id, timeStamp, text
  b. For owner_id, if(restrictedAccount) → Find u_id in restrictAccount → if found, update rComment=true for comment_id

  Return - Updated values

- Report:
  a. Create new entry in Report Log table and insert object_id and type(User/Post/Comment)

- Block Profile: User1 is blocking User2
  a. Insert User2_id in User's1 blockedUsers [ ] and remove User2_id from User1's followers [ ] and following [ ]
  b. Insert User1 in User2's blockedFrom [ ] and remove User1_id from User2's followers [ ] and following [ ]
  c. In user1_id Table of Chats database, find user2_id ⇒ if found, toRender=false. And then in user2_id Table of Chats Database, find user1_id ⇒ if found, toRender=false.

- Save/Unsave Post:
  a. Find p_id in Post then increment/decrement savedCount for p_id
  b. Find u_id in User Table then insert/remove p_id in/from savedPosts [ ]

- Follow/Unfollow profile: user 1 sent follow request to user 2
  a. Find User2_id → if(accountType=private) ? insert User1_id in User2's requests [ ] : insert User1_id in User2's followers [ ] and User2_id in User1's following [ ]
  b. if(request accepted) Find User2_id → insert User1_id in User2's followers [ ] and insert User2_id in User1's following [ ], then remove User1_id from User2's

requests [ ]
    c. if(request declined) Find User2_id → remove User1_id from User2's requests [ ]

- Follow/Unfollow Hashtag:
    a. Find hashtag in HashTag Table → if(present) return content of p_ids in posts [ ]
    b. Find u_id then update following [ ]

- Search fullname, username, place and hashtag:
    a. if(hashtag) ? Find hashtag in HashTag Table : Find search.value in User.username and username ⇒ if (present) ? return profilePicture, username, fullname, followers [ ] : Find search.value in Posts.location ⇒ if (present) ? return posts : nothing
    b. Find u_id in User table then insert user_id, user_id.profilePicture in searchHistory[ ]

- Logout:
    a. Find u_id in Users and remove session token[ ]

- Archive:
    a. Find p_id in Posts → update archived
    b. decrement postCount for u_id

- Change Password:
    a. if(forgot password selected) Find phone no/email/username in User Table's username/email/phone no
    b. if(forgot password selected) Trigger Send Email/SMS to user
    c. if(forgot password selected) Update the password and then go for step e
    d. if(change password selected) match value.old with password for u_id in User's Table⇒ if valid, update the password by value.new, else UI notification
    e. Update token [ ] for user

- Turn Commenting ON/OFF :
    a. Find p_id in posts and update commenting

- Change Profile Details Visibility:
    a. Find u_id then update accountType

- Delete Post/Story:
    a. Find p_id/s_id in posts/stories table then update deleted and modifiedAt and remove p_id/s_id from posts [ ]/stories [ ]
    b. If(post) → decrement postCount for u_id

- View Saved Posts:
    a. Find u_id then for savedPosts [ ] ⇒ find p_id in Posts Table → if(found && !archive || !deleted) ? return content of p_id : if(!found) ? remove p_id from savedPosts[ ] of u_id : nothing

- View post(s) a particular user or the user himself/herself is/are tagged in:
    a. For each p_id in taggedIn [ ] of user → find p_id then return content

- View posts i've liked:
    a. Find u_id ⇒ for each p_id in liked[ ] return content

- Archived Posts Log:
    a. Find u_id in User Table then for each p_id in posts[ ] ⇒ Find p_id in Posts Table then if(archived && !deleted) → return content of p_id

- Archived Stories Log:
    a. Find u_id in User Table then for each s_id in stories[ ] ⇒ Find s_id in Story Table then if(archived) → return content of s_id

- Temporarily Enable/Disable Account:
    a. Find u_id in User Table then match password, if(valid) continue, else terminate
    b. Update account and modifiedAt for u_id in User Table
    c. Find u_id in User Table then for each p_id in posts[ ] ⇒Find p_id in Posts Table then update disabled
    d. Find u_id in Comments Table then for each comment_id update disabled
    e. Find following [u_id] of user 1 and update followers of [u_id]
    f. Find followers [u_id] of user 1 and update followings of [u_id]
    g. Find u_id in User Table then for each s_id in stories[ ] ⇒Find s_id in Story Table then update disabled
    h. Find u_id in User Table then for each entry in taggedIn [ ] of u_id → find p_id in Posts and update tagged.type where tagged.u_id = u_id
    i. Chats
        i. Find set(to[u_id] U from[u_id]) U set(messageRequest[u_id])
        ii. Find user in all [u_id] and set toRender = False/True
    j. Find u_id in User Table then clear token [ ]

- View activity related to my account:
    a. Find u_id from logs and sort = -1
    b. Filter Tagged,comments,pending,mentions then return

- Login Activity Log:
    a. Find u_id in Log Table then filter Login and sort =-1 then return

- Request Account Verification:
    a. Create new entry in Request For Verification Table and insert u_id, image, fullname, knownAs, category

- Switch to Professional Account:
    a. Find u_id in User Table then update accountType2, accountType for u_id
    b. For each u_id in requests [ ], insert u_id in followers [ ] of user and user's id in all u_id's following [ ]

- Story Archive (Location):
    a. Find u_id in story where location!=NULL
    b. Return content of s_id

- View Entire History of Account:
    a. Find u_id in Logs Table and return all entries where activityRelated == false

- Delete Comment:
    a. Find comment_id && u_id in Comments table then delete entry from Comments table

- Comments' features:
    a. Find user_id in User Table then insert u_id in block_user_from_comment[ ]
    b. Update checkComment for user_id
    c. Update allowCommentsFrom for u_id in User Table
    d. Insert manual filter phrases/words in offensiveWords [ ] of user
    e. If (checkComment) → execute checkComment

- View and manage Search History:
    a. Find u_id in User Table then return seachHistory[ ]

- View and manage Recently Deleted Posts:
    a. Update deleted for p_id in Posts Table
    b. Find u_id && p_id in Posts then return modifiedAt, content of p_id where deleted == true

- Edit Posts:
    a. Find p_id in Posts then update caption/title/tagged/location, modifiedAt then return post

- Remove Follower: User1 removes User2
    a. Find User2_id in User Table then remove User1_id from User2's following[ ]
    b. Find User1_id in User Table then remove User2_id from User1's followers[ ]

- Hide/Unhide Like and View Counts:
    a. Find u_id in User Table then update likeViewCount

- Hide/Unhide Likes and Views count from a post:
    a. Find p_id in Posts Table then update count

- Allow tags from:
    a. Find u_id in User Table then update allowTagsFrom

- Manually Approved Tags:
    a. For p_id, check for each u_id in tagged [ ] → if(manuallyApprovedTags) ? insert p_id in toConfirmTag [ ]
    b. Execute approveTags trigger

- Hide story from particular user(s): User1 hides from User2
    a. Find User1_id then insert User2_id in hideStory [ ] of User1

- Remove Tag: User1 removes tag from User2's post
    a. Find user1_id in User Table then remove p_id from taggedIn [ ] of user1 and then execute removeTag trigger

- Allow @Mentions From:
    a. Find u_id in User Table then update allowAtMention
    b. When trying to mention people, execute this query (user1 mentions user2): find user2_id in User Table then return allowAtMention

- Close Friends: User1 adds/removes User2 to close friends
    a. Find user1_id in User table then insert/remove user2_id in closeFriends [ ] of User1

- Allow resharing of posts to stories:
    a. Find u_id in User Table then update allowToShareOnStory for u_id

- Manage Notifications:
    a. Find u_id in User Table then update manageNotification [key:value] of user

- View Insights(Professional/Business Account):
    a. Find s_id in Story Table then increment profileVisits/impressions/follows/forward/exitted/next counter in insights [ ] for

s_id
   b. Find p_id in Posts Table then increment profileVisits/follows/impressions counter in insights [ ] for p_id

- Mute stories: User1 mutes User2's stories
   a. Find user1_id in User Table then insert user2_id in muteStories [ ] of user1_id

- Mute posts and stories: User1 mutes User2's stories and posts
   a. Find user1_id in User Table then insert user2_id in muteStories[ ] and mutePosts [ ] of user1_id

- Add a story(old/current) to highlights:
   a. Find u_id in User Table then insert s_id in highlight[{ collectionName: s_id }] of u_id

- Restrict/Unrestrict Account: User1 restricts/unrestricts User2
   a. Find user1_id in User table then if(!restrictedAccount) update restrictedAccount=true, then insert user2_id in restrictAccounts [ ] of user1_id

   b. When posting a comment by user2_id on user1_id's post, find user1_id in User Table then if(restrictedAccount) find user2_id in restrictAccounts [ ]. If found, rComment of comment_id = true.

   c. If unrestricting ⇒ Find user1_id in User table then if(restrictAccounts.size()==1) restrictedAccount = false, then, remove user2_id from restrictAccounts [ ] of user1_id

- Polls:
   a. Find s_id in Story then insert new entry in props.polls with {user_id : boolean} ⇒ boolean will be true if right, else false

- Questions:
   a. Find s_id in Story then insert new entry in props.questions with { user_id : response(text) }

- Add location in story:
   a. Create new entry in s_id then insert content, location, timeStamp, u_id, viewType, mention[ ] (if any), music(if any), link(if any) and props(if any) for it

- Mention other users on story:
   a. Create new entry in s_id then insert content, user_id(s) in mention [ ], timeStamp, u_id, viewType, location(if any), music(if any), link(if any) and props(if any) for it

- Emoji Slider:
    a. Find s_id in Story then insert new entry in props.eSlider with { user_id : int }

- Countdown:
    a. Find u_id in user Table then insert new entry in countdown [ ]

- Quiz:
    a. Find s_id in Story then insert new entry in props.quiz with { user_id : response }

- See who and how many saw my story:
    a. Find s_id in Story then return viewedBy [ ]

- Add music on story:
    a. Create new entry in s_id then insert content, music, timeStamp, u_id, viewType, mention[ ] (if any), location(if any), link(if any) and props(if any) for it

- Hide/Unhide a post you're tagged in from the tagged posts section on "My Profile page":
    a. Hide ⇒ Find u_id in User Table then remove p_id from taggedIn [ ]
    b. Unhide ⇒ Visit the post then select "Show on My Profile" ⇒ Find u_id in User table then insert p_id in taggedIn [ ]

- View comment/post liked by:
    a. Post ⇒ Find p_id in Posts then return likedBy [ ]
    b. Comment ⇒ Find comment_id in Comment then return likedBy [ ]

- View all posts with a hashtag:
    a. Find hashtag in HashTag then for each p_id in posts [ ] → return content of p_id

- View followers/following of a user: user1 views user2's followers/following
    a. If(accountType==public) Find user2_id in User Table then return following [ ] / followers[ ]
    b. If(accountType==private) Find user2_id in User Table then if(followers.find(user1_id) != -1) ? return following [ ] / followers[ ] : return 400 Bad Request

- Add/remove a post to/from a guide:
    a. Find u_id in User Table then insert p_id in guides[{ collectionName: p_id }] of u_id

- Remove IGTV/Reel from profile grid:

    a. Find u_id in User Table then insert p_id in hideFromProfileGrid[ ]

- Delete Account:
    a. Find u_id in User Table then account = disable
    b. Create new entry in Delete Account Table then insert u_id

Additional Queries:

- Share(Post/Profile/Story): User 1 is sharing to user 2
    a. In Chats database, in User1_id Table, find User2_id, if(not present) ? create new entry then Insert from(user1_id), to(user2_id), data(link=>get request) : Insert from(user1_id), to(user2_id), data(link=>get request)
    b. if(post/story) increment sharedCount for p_id/s_id
    c. If(Profile & owner's accountType=private) return username, fullname, profilePicture
    d. If(Profile & owner's accountType=public) return profilePicture, username, fullName, posts.content(latest 6)
    e. if(Post & owner's accountType=public) return content, owner_id's profilePicture, owner_id's username
    f. if(Post & owner's accountType=private) → Find owner's followers[user2_id]
    g. If present, content, owner_id's profilePicture, owner_id's username
    h. If not present return 400 (Bad Request)
    i. if(Story & owner's accountType=public) return content of s_id
    j. if(Post & owner's accountType=private) → Find owner's followers[user2_id]
    k. If present return content of s_id
    l. If not present return 400 (Bad Request)

- Download Account Data:
    a. Find u_id in Posts, Comments, Story, User, Logs then return
    b. Return u_id from chats
    c. Return User.groups[ ] from Broadcast

- View Active/Inactive:
    a. Find u_id in User Table then update activity for u_id
    b. Chats
        i. To be handled by FrontEnd

- Surf Explore Section:
    a. Find post { }

- Chats : User1 sends message to User2, User1 is added/removed to/from group by User2
    a. Create new entry in user1_id of chats for user2_id then insert data, to(User2_id) then create new entry in user2_id of chats for user1_id then insert data,

from(User1_id)

b. if(!(accountType==public || (user2_id.followers[user1_id] || user2_id.following[user1_id]))) find user2_id in User Table then insert user1_id in messageRequests[ ]
c. Find user1_id in User table and if(accountType==public) or followers[user2_id] → Insert/delete {name : g_id} to/from groups[ ] of user1_id, else, Insert g_id to messageRequests[ ] of user1_id

- Allow Replies and Reactions on stories: User1 reacts/replies on User2's story
    a. Find u_id in User Table then update allowReplyReaction for u_id
    b. if(allowReplyReaction==All) Find/Create entry in user2_id of chats for user1_id then insert data, from(User1_id) and create/find new entry in user1_id of chats for user2_id then insert data, to(User2_id)
    c. if(allowReplyReaction==following) Find User2_id in user Table then find User1_id in User2_id's following [ ]. If (present) ? Find/Create entry in user2_id of chats for user1_id then insert data, from(User1_id) and create/find new entry in user1_id of chats for user2_id then insert data, to(User2_id) : return 400 Bad Request
    d. if(allowReplyReaction==none) return 400 Bad Request

- Allow sharing of stories: User1 shares story of User2 with UserX
    a. Find u_id in User Table the update shareStory for u_id
    b. Find user2_id in user table then if(shareStory) ? Find/Create entry in user1_id of chats for userX_id then insert data, to(UserX_id) and create/find new entry in userX_id of chats for user1_id then insert data, from(User1_id)
    c. Find s_id in Story then increment sharedCount for s_id

- View Story/Live:
    a. Find s_id in Story then return timeStamp, content, type, viewType, u_id, mention[ ], location, music, link

- Live:
    a. After ending live, Find u_id in User table then if(liveArchive) create new entry in LiveVideos then insert Video, u_id, timeStamp

- View Profile:

    Find u_id in the User Table then if(account == enabled) ? (if(blockedUsers.find(user_id) == -1) ? (return 400 Bad Request : if(accountType==private) ? find user_id(visitor) in followers[ ] of u_id → if(not present) return profilePicture, bio, following[].size(), followers[ ], postCount, fullname, username) :: return profilePicture, bio, following[].size(), followers[ ], postCount, fullname, username, taggedIn [ ], posts[ ], hideFromProfileGrid[ ]; execute story)

- View Post:

  Find p_id in Post Table then if(disabled || archived || deleted) return NULL : { (return p_id, u_id, createdAt, modifiedAt, caption, location, tagged[u_id : type], type(Reel/Post/IGTV), likes(counter), likedBy[u_id], savedCount, hashTags[ ], deleted(boolean), commenting(boolean), archived(boolean), count{Hide/Unhide}(boolean), title, content[ ], music, sharedCount, disabled(boolean), insights[(profileVisits, follows, impressions) ⇒ int]) && find u_id in User Table then return following[ ], allowToShareOnStory && find p_id in Comment Table then if (disabled) ? return NULL : { return p_id, u_id, likes(count), likedBy[u_id], timeStamp, rComment(boolean), comment_id, text, owner_id, bComment(boolean) } }

- View Story:

  For each u_id in following [ ], if(disabled) ? return NULL : (each s_id in stories[ ] → find s_id in Story then if(viewType==closeFriends) ? (Find u_id in User Table then return u_id.closeFriends[ ], u_id.allowReplyReaction, u_id.hideStory[ ], u_id.shareStory, s_id.content, s_id.type) : return u_id.allowReplyReaction, u_id.hideStory[ ], u_id.shareStory, s_id.content, s_id.type)

- View Chats:

  - Return username Table from Chats database

    Here, u_id (Requester)
  - If (messageRequest==decline) ⇒ add u_id in declinedUsers [ ]
  - For all g_id in groups[ ], return all tables

- Delete Account:

  - For each entry in the table, Find u_id in user table then for each p_id/s_id in posts[ ]/stories[ ], find p_id/s_id in Post/Story Table then remove the entire entry
  - Chats
    - Find set(to[u_id] U from[u_id]) U set(messageRequest[u_id])
    - Find user in all [u_id] and delete entry
    - Drop table for u_id
    - Remove u_id for all g_id in u_id.groups[ ] from group[ ]