Java Project Report:
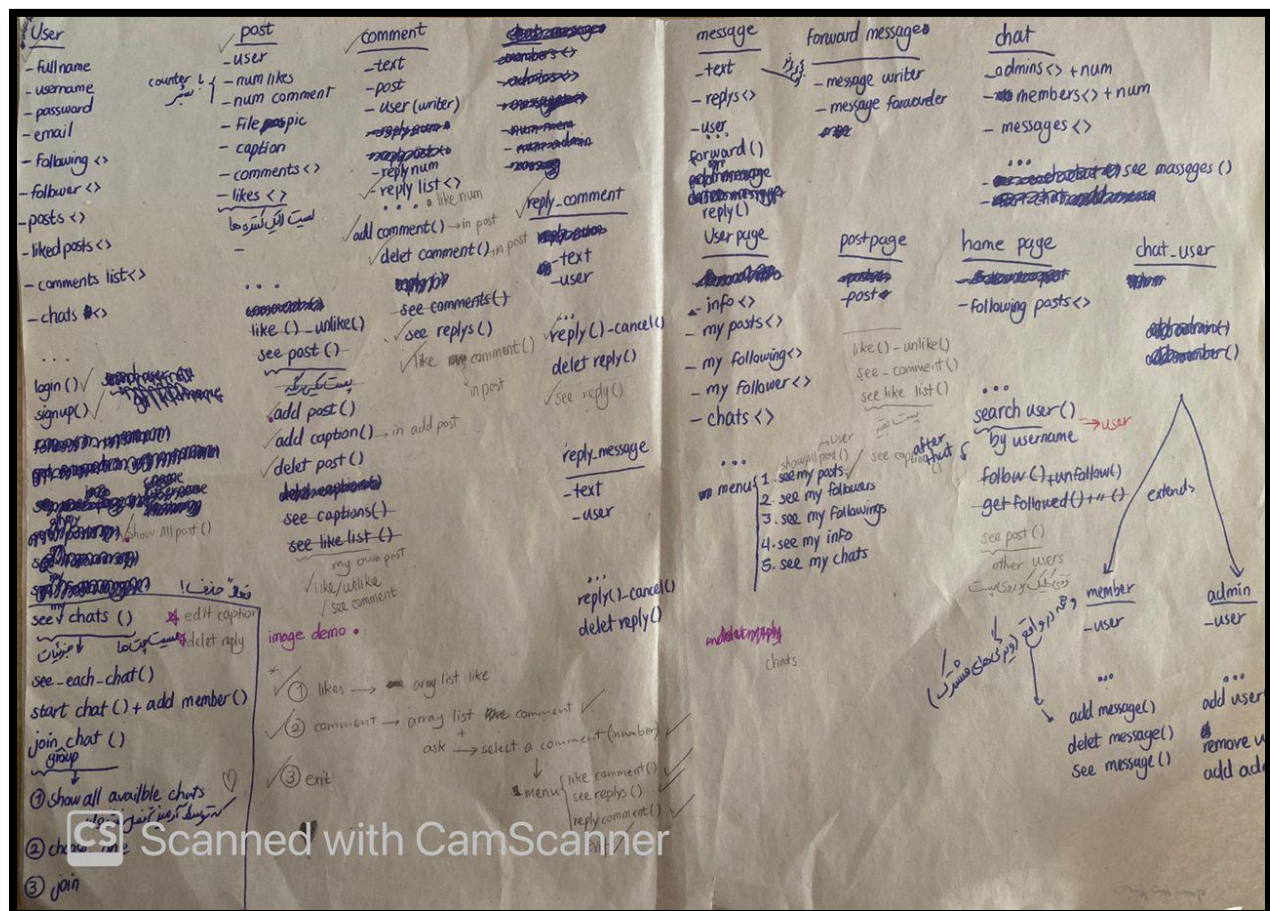
A Social Media Application

Team Members: Soroush Abdollahi – Pardis Basiri – Nadia JavanMardi

Date: January 23

UML Design)

Our first UML Design was on paper and changed a couple of times over discussions:

Then we defined the process that starts with user's logging, to clear exactly how many methods we need and how they should work.

① sign up / login

↓ Age

sign up { full name (nickname)
username
password
email
phone number
bio
profile picture / Email Var

see like → ~~Hem~~ like num + list liked user
see comment → # comment I: { user:
text
like num
like it
reply it
see # replies → { user
text

~~2222~~ ↓

~~uuuuu~~ main menu
~~uuuuu~~ (home) Page

see ~~pooxxx~~ my user page

{ my info · choose a post or back post page { user:
my posts   text:
my followers:∅ my following: 0   1. like ~~num~~
                                            2. comment ~~xxxx~~
select user   ~~xxxx~~
                                            see like
                                            like / reply / back ← see comm~~

see my home page → اگر تویی ~~رو پرنویی~~ لو following سا ~~نیست~~ ✳
                    ۵ ایش ~~فروشن~~ تر پرستن بد
                                            post page

see my chat page { start a new chat
                   ~~seemxxx~~ old chats ~~xxxx~~ سیس پویکسین
                                          اریتن user لیس ریتی
                                          اسل .

select { see posts ← search a user
user   { follow   post
       { see info   ↓
                   { text :

Enter a name for chat
                              other
* start new chat → { ~~xxxx xx xxxxxx~~ Enter admins username
                    " ≈ memebers "
                    send a message if you want

All chats page
old chats → list chat names → ~~message~~ page { chat name:
                                                last 10 messages
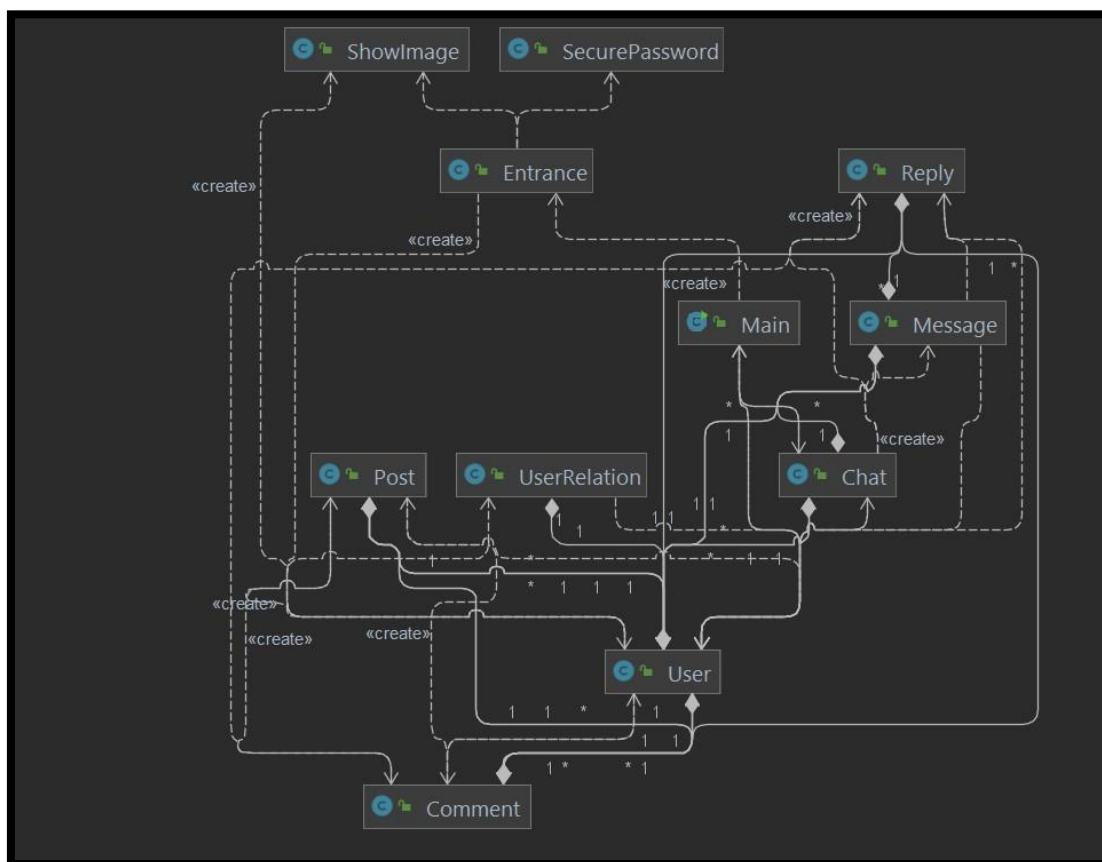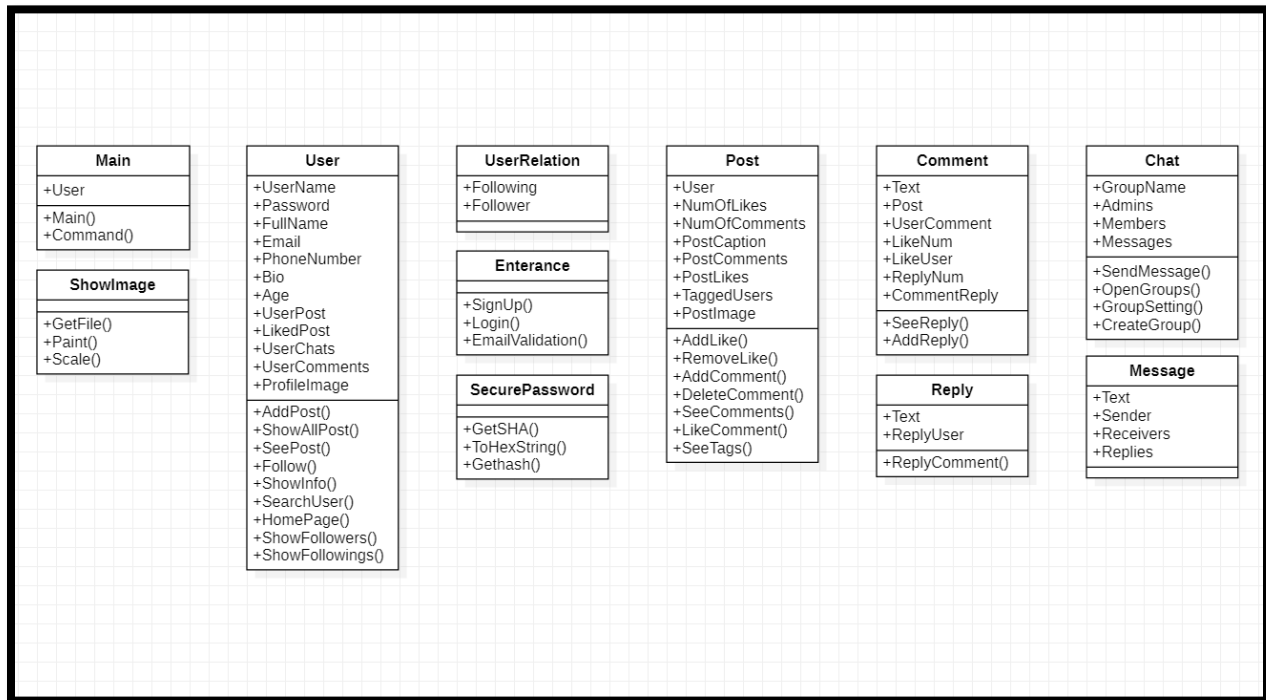   { sender:
                                                { text

                                                select a message ~~xxxx~~
                                                or back ↓
                                                forward
                                                reply

send message
~~xxxx~~ ~~xxxxx chat name~~
chat name:

After all these steps we came up our final UML design:

Main
+User
+Main()
+Command()

ShowImage
+GetFile()
+Paint()
+Scale()

User
+UserName
+Password
+FullName
+Email
+PhoneNumber
+Bio
+Age
+UserPost
+LikedPost
+UserChats
+UserComments
+ProfileImage
+AddPost()
+ShowAllPost()
+SeePost()
+Follow()
+ShowInfo()
+SearchUser()
+HomePage()
+ShowFollowers()
+ShowFollowings()

UserRelation
+Following
+Follower

Enterance
+SignUp()
+Login()
+EmailValidation()

SecurePassword
+GetSHA()
+ToHexString()
+Gethash()

Post
+User
+NumOfLikes
+NumOfComments
+PostCaption
+PostComments
+PostLikes
+TaggedUsers
+PostImage
+AddLike()
+RemoveLike()
+AddComment()
+DeleteComment()
+SeeComments()
+LikeComment()
+SeeTags()

Comment
+Text
+Post
+UserComment
+LikeNum
+LikeUser
+ReplyNum
+CommentReply
+SeeReply()
+AddReply()

Reply
+Text
+ReplyUser
+ReplyComment()

Chat
+GroupName
+Admins
+Members
+Messages
+SendMessage()
+OpenGroups()
+GroupSetting()
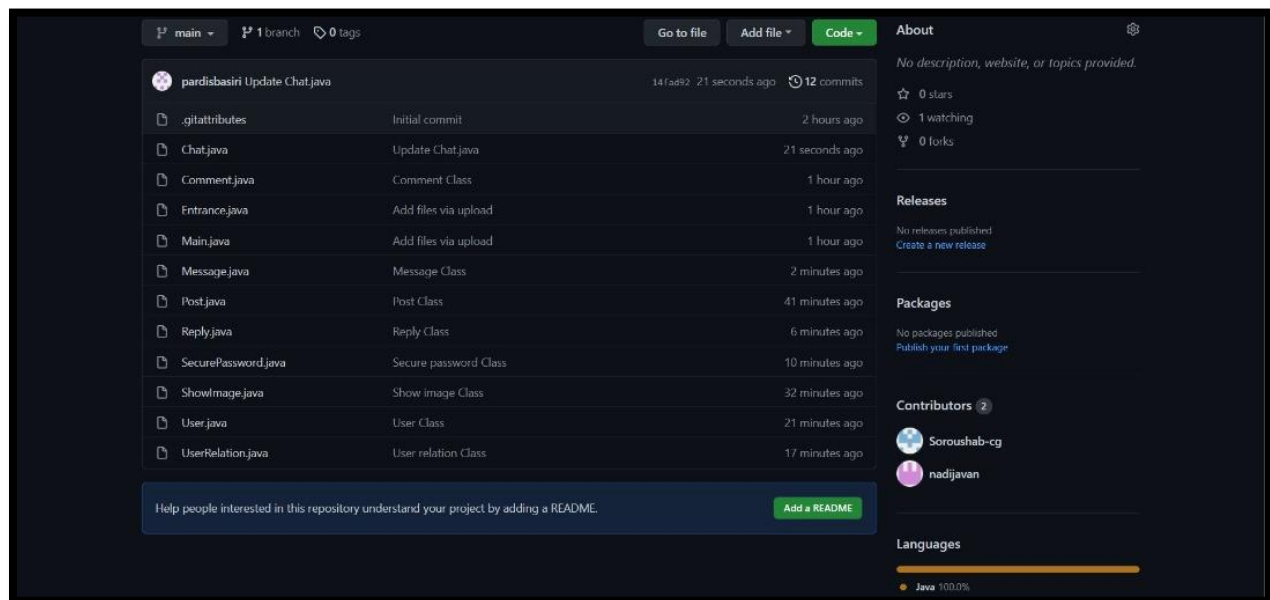+CreateGroup()

Message
+Text
+Sender
+Receivers
+Replies

GitHub)

We choose GitHub for our cloud git repository and all members contribute.

Name: Java_Project

Link: https://github.com/Soroushab-cg/Java_Project

Member's GitHub: Soroushab-cg // pardisbasiri // nadijavan

Database Management Strategies)

Our project is not connected to any database but here is our progress in this area:

We needed a database to store data and our choice was H2 because:

- It is free
- It is light weighted
- Use very less memory
- Doesn't need to install hash to database

Then we decided to use it's zip file version cause :

- It can operate on any operating system
- It is independent
- It doesn't need installation

At this point we had to choose one of the three modes :

- In memory
- Embedded
- Server mood

In memory was good for testing, it didn't need physical but we got error when we connected database after disconnecting it once in this mood, so our choice was Embedded mood for testing. And for our final choice we decided to go with server mood because in this mood we can connect with a number of clients and it is permanent with a physical memory and it needs a data base file.

Important feature:

- This project is functional so it very clean and understandable.

- One of the special methods which improved reusability in our code, is Command. We used it anywhere we needed a menu thing without repeating same switch cases and sentences.

- Instead of having a very messy User class we brought Entrance part and User Relation (Following/Follower) part into two different classes.

- For better service we provided an email validation system and age limit of 12 to make sure right users are using it.

- For user's security we turn password to hash code.

- For better access in our main menu we have home page, user page and chat page.

- In additional features users can tag each other on their posts and see tagged people, users can reply to comments or like comments.

- In chat section users can have saved message by making a new chat and adding no one else. Users can have private message with other users by just adding them to the chat and they can make group chat and send message to more than one user at a time. Users can have admin or member access and they can send message as well as replying to a message or forwarding it.

- Users can block each other and they won't be able to find each other on search part and the blocked user won't be able to see the profile of the other user or add him/her to chat. But the user how blocked can find the blocking user on the list and unblock him/her.