پروژه فاز صفر

Object Oriented Programing

برنا خدابنده

بیان ملزومات اجزای پروژه شبیه سازی یک شبکه اجتماعی

مهندسی برق

۲۲ فروردین ۱۴۰۱

فهرست مطالب

۱ کاربران

ابتدا برای کنترل کردن اطلاعات و ذخیره کردن آنها و انجام عملیات های بین هر کاربر نیاز به کلاس خود کاربر داریم.

†User classes

```
public abstract class User {
    public String username, firstName;
    public boolean isPublic;

    public void sendMessage(Message message, Chat chat) {
        // will send message to the desired chat
    }
    public abstract void Post(Post post);
}

```

Source Code 1: abstract User class

```
public class normalUser extends User {
    public String lastName;

    public normalUser(String username, String firstName, String
    lastName, boolean isPublic) {
        // only called when a user registers
    }
    public void Post(Post post){
        //posts a Post to the dataBase
    }
}

```

Source Code 2: normalUser class

```
public class businessUser extends User {
    public String businessType;

    public businessUser(String username, String businessType,
   boolean isPublic) {
        // only called when a user registers
    }

```

```
8       public void Post(Post post) {
9           // posts a Post to the dataBase and handles promotions
10          // specialized for business Users
11      }
12
13      public void returnEngagementStats() {
14          // gives detailed info on how the business is
        performing
15      }
16  }
17
```

Source Code 3: businessUser class

کلاس businessUser وارث کلاس User بوده و مشابه با normalUser فقط برای businesUser تفاوت هایی در gui و همچنین اینظور که متد هایی مانند پست کردن برای businessUser خاص تر بوده و همچنین promotion و چیز های مشابه را انجام میدهند تا برای بیزنس ها مناسب تر باشند [1]
همچنین وجود متد های خاص برای این نوع کاربر که اطلاعات مهم فراتر از یک کاربر عادی را به این نوع کاربر میدهد.

[1] متد ها و اطلاعات لازم برای promotion جدا توضیح داده میشوند

۲ شبکه کاربران

حال برای اینکه اطلاعات مربوط به همگی کابر ها را بتوانیم کنترل و ذخیره کنیم و یا احراز هویت یک کاربر، نیاز به یک table داریم که در آن همه اطلاعات ذخیره شوند مانند:

- نام کاربری(primary)
- رمز عبور(encrypted)
- زمان ثبت نام
- نوع اکانت
- visibility
- تعداد پست/following/follower
- اطلاعات نهان مانند promotionIndex و اندیس هایی برای انتخاب محتوای نشان داده شده

user_id	pass_enc	date joined	type	visibility	followers	...
borna._.kh	jijaspmx24x	۲۲ فروردین ۱۴۰۱	basic	public	512	...
sep_h	oppoa124k	۲۲ فروردین ۱۴۰۱	basic	public	12	...
sepronites.co	mdpsapowkx	۲۲ فروردین ۱۴۰۱	business	public	10012	...
hoss_anji	fj120cxaxcl	۲۲ فروردین ۱۴۰۱	basic	private	2	...

Table 1: user info table

۱.۲ احراز هویت و encryption

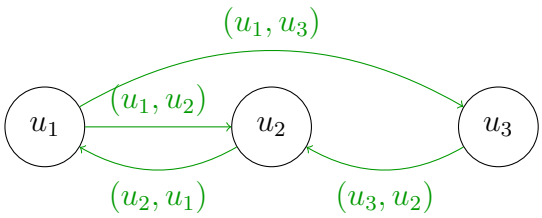
برای احراز هویت کاربران و همزمان حفظ privacy و امنیت کاربران در دیتابیس پسوورد اصلی کاربر ها را ذخیره نمیکنیم بلکه طبق یک الگوریتم ثابت و برگشت ناپذیر حالت رمزگذاری شده از پسوورد را ذخیره میکنیم و در مرحله احراز هویت پسوورد را از کاربر گرفته و چک میکنیم که بعد از رمزگذاری با اطلاعات داخل دیتابیس یکی باشد.[۲] [۳]

[۲] ممکن است از captcha نیز استفاده شود
[۳] الگوریتم های مورد نیاز در لیست کد ها توضیح داده شده اند

۳ ارتباطات بین کاربری

۱.۳ دنبال کردن

هر کدام از راس های u_i .نشان دهنده یک کاربر هستند و ارتباط های بین آنها نشان دهنده دنبال کردن یا شدن بین کاربر ها است
به صورتی که یک خط بین u_i و u_j به معنای این است که کابر i کاربر j را دنبال کرده.



$(u_1, u_2),(u_1, u_3),(u_2, u_1),(u_3, u_2)$

[۴]

میتوان این ارتباطات را به صورت یک table نیز نشان داد.
با استفاده از یک table با ۲ تا primary که یکی از آنها follower_id و دیگری followed_id میباشد.
در نتیجه هر سطر نشان دهنده یک ارتباط منحصر به فرد خواهد بود.

follower_id	followed_id	date followed	type
u_1	u_2	۲۲ فروردین ۱۴۰۱	basic
u_1	u_3	۲۲ فروردین ۱۴۰۱	vip
u_2	u_1	۲۲ فروردین ۱۴۰۱	basic
u_3	u_2	۲۲ فروردین ۱۴۰۱	basic

Table 2: user connection table

کلاس استفاده شده برای رسیدن به این table و متد ها:

†Connections class

```
public List<Integer> getFollowers(){// returns user_id of
    followers
}
public void follow(User user){// adds user to followers DB
}
public List<Integer> getBlockedList(){
```

[۴] متن های نارنجی خلاصه شده ارتباطات بین کاربر ها هستند

```
}
public void block(User user){// adds user to blocks DB
}
public List<Integer> getMutedList(){
}
public void mute(User user){// adds user to blocks DB
}

```

Source Code 4: connections

۲.۳ بلاک کردن

برای بلاک کردن نیز از table های مشابهی استفاده میکنیم فقط در داخل خود کلاس ها و نتایج سرچ userVisibility را false میگذاریم.
مشابه برای mute کردن. ممکن است چنین کلاس هایی برای سادگی کار تعریف شوند:

†Optional classes

```
public class followers implements table{

}
public class blocked implements table{

}
public class muted implements table{

}

```

Source Code 5: connections

۴ پیام رسانی

۱.۴ کلاس و متد های مربوط

برای این منظور نیاز به ذخیره پیام ها ، تضمین ارسال درست و محرمانه پیام ها و نمایش درست آنها هست.
برای خود پیغام از کلاس message استفاده خواهیم کرد.
و برای مکانی که ارسال میشوند در نظر میگیریم که هر پیام در فضای یک chat ارسال میشود پس کلاس chat را نیز تعریف میکنیم

†Messaging classes

```
public class Message {
    public long id, date; // unique identifier and the date the
     message was sent
    public String message;
    public Image messageImage;// (Optional) null for text based
     messages
    public long inReplyToUserID; // (Optional) the id of the
    user that it was a reply to, null for original posts
    public User sender; // user that sent the message
    public Chat chat; // the chat that the message has been
    sent into
    private int decryptionKey; // used to support end-to-end
    encryption, not saved in the dataBase

    public Message() {// used when user sends a message
    }
}

```

Source Code 6: Messages

```
public class Chat {
    public long id;// unique identifier
    public String type;// can be pv/group or channel
    public String name; // name displayed on top of chat
    public Image chatPhoto;// picture displayed for chat
}

```

Source Code 7: unique chats

برای ارسال هم پیام هم از متد ای مانند

```
public void sendMessage(Message message, Chat chat);
```

استفاده خواهیم کرد که پیام مورد نظر را توسط User به چت مورد نظر فرستاده و داخل دیتابیس ذخیره میکند(احتمالا در روند اجرا چند کلاس private دیگر به منظور ارسال پیام با فورمت های مختلف نیز استفاده شود)

۲.۴ دیتابیس پیام

برای ذخیره پیام ها و اطلاعات مربوطه، در دیتابیس یک table ایجاد میکنیم با متغیر اصلی ،msg_id محتویات پیام، ارسال کننده پیام و chat ای که داخل آن فرستاده شد.

msg_id	message_enc	sender	chat_id	reply_id
1	**********	borna._.kh	12425	null +-
2	**********	sep._.h	12425	2
3	**********	hoss_anji	12425	41
4	**********	mo.makk	19050	3

۱.۲.۴ message_enc

برای ذخیره محتوای پیام و حفظ privacy کاربران بجای ذخیره پیام اصلی یک پیام رمزگذاری شده را ذخیره میکنیم.
که پیام ها و فرایند ارسال پیام end to end encryption داشته باشند

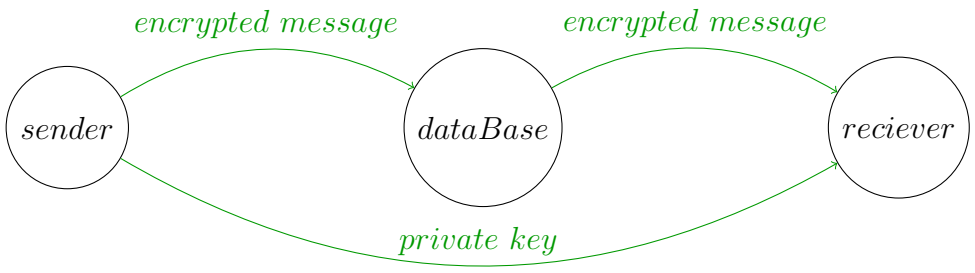


Figure 1: messages are encrypted using the private key

۳.۴ ساخت گروه

برای پیام رسانی کلی و برای اینکه گروه بسازیم باید ذخیره کنیم که یوزر ها هرکدام مجاز ارسال پیام به کدام چت ها هستند.

۱.۳.۴ متد های نیاز

اطلاعات مورد نیاز شامل تعداد ممبر ها، اعضای گروه و غیره بوده و همجنین متد هایی نیاز داریم برای ادد کردن/پاک کردن افراد از یک چت بوده
همچنین نیاز داریم بتوانیم توانایی های اعضا را کنترل کنیم مانند قابلیت ارسال پیام در آن چت

†chat handling methodes

```
1 public void addToGroup(User user){// adds user to the chat
2 }
3
4 public void removeFromGroup(User user){// removes user from
      chat
5 }
6
7 public void changePermissions(User user){// handles permissions
      , might take extra input
8 }
9
```

۲.۳.۴ دیتابیس

از متغیر اصلی chat_id استفاده کرده و طبق آن بقیه اطلاعات مربوط مانند user_id اعضای چت و permissions سیو میکنیم.
قابل توجه است که برای هر ۲ یوزر یک چت در نظر گرفته شده است از تایپ .private

chat_id	user_id	permissions
1241907	borna._.kh	11
1490163	sep_h	01
912490	hoss.anji	00
129041	mo.maccintosh	11

۵ پست

۱.۵ کلاس و متد های مربوط

برای ذخیره پست ها و کنترل کردن آنها توسط کاربر یک کلاس برای هر پست در نظر میگیریم.
هر کامنت زیر هر پست را نیز به صورت یک پست در نظر میگیریم در نتیجه در کلاس پست یک سری پارامتز برای اطلاعات در مورد اینکه اگر ریپلای است به کدام کاربر و کدام پست هست ذخیره میکنیم.

برای نشان دادن درست صفحه اول به کاربر یک کلاس feed مربوط به صفحه اصلی نیز در نظر میگیریم.

†Posting classes

```
public class Post {
    public long id;// unique identifier for each post
    public long inReplyToPostID; // (Optional) the id of the post that it was a reply to, null for original posts
    public long inReplyToUserID; // (Optional) the id of the user that it was a reply to, null for original posts
    public String text;
    public Image picture; // (Optional) is null for text posts
    public User sender; // user that posted it
    public int likes;// number of likes on the post
    public int views;// and other stats that show traction and are'nt publicly visible
    public long date;

    public Post() {// used when Users make a Post
    }
}

```

Source Code 8: Post class

```
public class feed {
    Set<Post> postsSet = new TreeSet<Post>();// set of all posts the user should see

    public void refreshFeed() {// finds new posts to show on the feed
    }
}

```

Source Code 9: Post class

۲.۵ دیتابیس پست ها

برای ذخیره پست ها و اطلاعات مربوطه، در دیتابیس یک table ایجاد میکنیم با متغیر اصلی post_id، محتویات پست، ارسال کننده پست و اطلاعات در مورد اینکه پیام به کجای رشته پست ها قرار دارد با استفاده از reply_id و اطلاعات مربوطه مانند تعداد لایک و سایر داده ها شامل تعداد ویو، engagment ، درصد کلیک ، و سایر اطلاعات مربوط به promotion برای پست.

post_id	post_data	sender	reply_id	likes	...
1	**********	borna.__.kh	12425	125	...
2	**********	sep.__.h	null	11	...
3	**********	hoss_anji	12425	2	...
4	**********	mo.makk	19050	51052	...

Table 3: posts DataBase

برای ساخت ، پاک کردن و ادیت کردن یک پست در داخل کلاس یوزر ها از متد های

```
public abstract void Post(Post post);//has specialized
//implementations for normalUser and businessUsers

public void deletePost(Post post) {// deletes the post
    }
    string hi = "salam string";

public void editPost(Post post, Post newPost) {// edits the
   post
}

```

Source Code 10: Post class

استفاده خواهیم کرد که هم یک آبجکت پست تولید یا ذخیره کرده و یا تغییر میدهد. و هم اطلاعات لازم را در دیتابیس ذخیره میکند.

۶ لایک

۱.۶ متد های نیاز

لایک کردن یکی از اصلی ترین خاصیت های این برنامه خواهد بود در نتیجه حایز اهمیت هست که به درستی هندل شوند.
برای ذخیره لایک ها از این ۲ متد استفاده میکنیم درکلاس User و همجنین درکلاس Post برای تضمین یک متد قرار میدهیم. [۵]

```
public void likePost(Post post){ // adds user to the likes DB
}

public void unlikePost(Post post){ // removes user from likes
    DB
}
"this method below is from the Post class and not from the User
     class"
public void validateLikes(){
    //counts likes from the data base and validates number of
    likes
}


```

Source Code 11: Post class

۲.۶ دیتابیس

کافیست در table لایک ها ۲ مورد post_id و user_id را ذخیره کنیم.
هر سطر در نتیجه با این ۲ متغیر اصلی یک لایک یکتا را مشخص میکند

post_id	user_id
12414	borna._.kh
12414	sep_h
20133	makk.ion
12334	sepro.co.ir

Table 4: likes Database

۶

[۵] دیسلایک ها به شکل مشابه هندل میشوند
[۶] ممکن است یوزرنیم یا id انتخاب شود اهمیت در یکتا بودن این پارامتر برای هر کاربر است

۷ پیشنهاد محتوا

باید یک متود ای تعریف کنیم که بتواند با یک عدد میزان علاقه شخص به یک پست را سنجش کند و سپس پست ها را توسط این میزان سورت کنیم و به این ترتیب در صفحه اکسپلور شخص نشان دهیم.

۱.۷ روش استفاده شده

برای این میتوانیم از داده هایی مانند پست هایی که افرادی که شخص دنبال میکند لایک یا کامنت میکنند و یا با در نظر گرفتن چندین کتگوری که یوزر به آنها علاقه دارد و سنجس تطابغ پست با این کتگوری ها. اینگونه که برای هر یوزر یک میز از ضرایب برای مقداری که یک کتگوری را میپسندد را ذخیره میکنیم و بعد مقدار اولیه علاقه به پست را تایین میکنیم به صورت که q_i مقدار تطابغ با کنگوری و V_i میزان علاقه فرد به آن کتگوری است:

$$\Sigma q_i V_i = L_0$$

پارامتر دیگر که مقدار علاقه افرادی که دنبال کرده ایم را نشان میدهد که l_i نشان میدهد که آن فرد لایک کرده است یا نه و S_i میزان نزدیکی ما به آن فرد است که با نسبت اشتراک پست های لایک شده محاسبه میشود

$$\Sigma l_i S_i = L_1 \ , \ L = L_0 + L_1$$

[۷] در آخر از پارامتر L برای سنجش پست ها استفاده خواهیم کرد

```
1 public int likabillity(Post post){//returns L value for said
    post
2 }
3 "in the explore page class"
4 private void sortPosts(){//will sorts the List of posts
    according to the L values
5 }
6
```

تمامی این مقادیر را در یک تیبل مخصوص با متغیر های اصلی post_id و user_id همراه با likabillity آن پست ذخیره کرده و مرتب میکنیم.

[۷] این فرمول ها احتمال تغییر بسیار دارند و این یک ساده سازی اولیه است

۸ رمزنگاری

برای این منظور از الگوریتم هایی استفاده میکنیم که اطلاعات را به اطلاعات غیر قابل دسترسی تبدیل کنیم برای امنیت بیشتر و همجنین محرمانه تر بودن اطلاعات.

۱.۸ پسوورد

برای این از یک تابع رمزنگاری بدون کلید استفاده خواهیم کرد به صورتی که روشی برای تبدیل پیام رمزنگاری شده به پیام معمولی وجود نداشته باشد.

```
1  public class crypt {
2      public static byte[] getSHA(String input) throws
       NoSuchAlgorithmException {
3          MessageDigest md = MessageDigest.getInstance("SHA-256")
       ;
4          return md.digest(input.getBytes(StandardCharsets.UTF_8)
       );
5      }
6
7      public static String toHexString(byte[] hash) {
8          BigInteger number = new BigInteger(1, hash);
9          StringBuilder hexString = new StringBuilder(number.
       toString(16));
10         while (hexString.length() < 32) {
11             hexString.insert(0, '0');
12         }
13         return hexString.toString();
14     }
15
16     public static String encryptedString(String input) throws
       NoSuchAlgorithmException {// actual method used
17         return toHexString(getSHA(input));
18     }
19 }
20
```

۱.۱.۸ salting

در دیتابیس اصلی که اطلاعات کاربری با ذخیره میکنیم. برای هر کاربر یک مقدار به عنوان salt نیز ذخیره میکنیم و در فیلد پسوورد ما مقدار $crypt.encryptedString(password + salt)$ را ذخیره کرده

و با همین تابع مقایسه میکنیم اینگونه برای دو کاربر با پسوورد یکسان هم مقادیر متفاوتی در دیتابیس ذخیره میشود. salt ها موقع ساخت اکانت به صورت استرینگ رندوم تولید میشوند

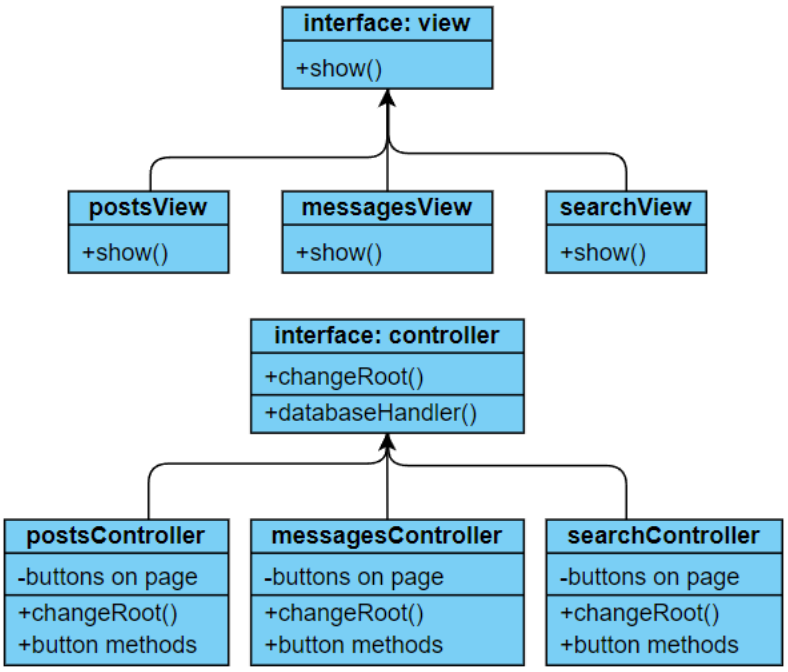
۲.۸ پیام ها

برای اینکه نیاز به بازیابی پیام ها هست در این رمزنگاری از یک تابع کلید دار استفاده خواهیم کرد. هر پیام یک private_key دارد این کلید ها در دیتابیس ذخیره نشده و بجای آن در حافظه دستگاه هر کاربر بطور جدا ذخیره میشوند و توسط پسوورد هر کاربر رمزنگاری میشوند. [۸]

[۸] همچنین میتوان انکریپت شده های $private\_key$ را در دیتابیس اصلی ذخیره کرد که از لحاظ امنیتی بدتر است ولی برای استفاده از چند دستگاه و مشابه آن کار را راحت تر میکند

۹ صفحه های مختلف

برای هر صفحه یک کلاس view و یک کلاس controlller میسازیم [۹]



شکل ۲: viewers and controllers

viewer

برای نمایش اطلاعات مورد نیاز استفاده میشود.

controller

برای گرفتن اطلاعات از کاربر و انجام تغییرات مناسب در اطلاعات نمایش داده شده، تغییر بین صفحه ها و کنترل کلی اتفاقات استفاده میشود

[۹] در شکل فقط ۳ تا از این کلاس ها نام برده شده ولی در واقع برای تمامی صفحه های اپ چنین کلاس هایی خواهیم داشت

۱۰ شرح دیتایس

۱.۱۰ table های مورد نیاز

- کاربران و اطلاعات کاربری
- شبکه لایک و دیسلایک
- پست ها و اطلاعات پست ها
- کتگوری ها و امتیاز پست ها در هر کتگوری
- شبکه فالوور ها/بلاک ها
- پیام ها و اطلاعات مربوطه

۲.۱۰ روش ذخیره اطلاعات