BASIC TWITTER

Sagar Kalsaria Ritik Roongta Yash Jain

170050006 170050056 170050055

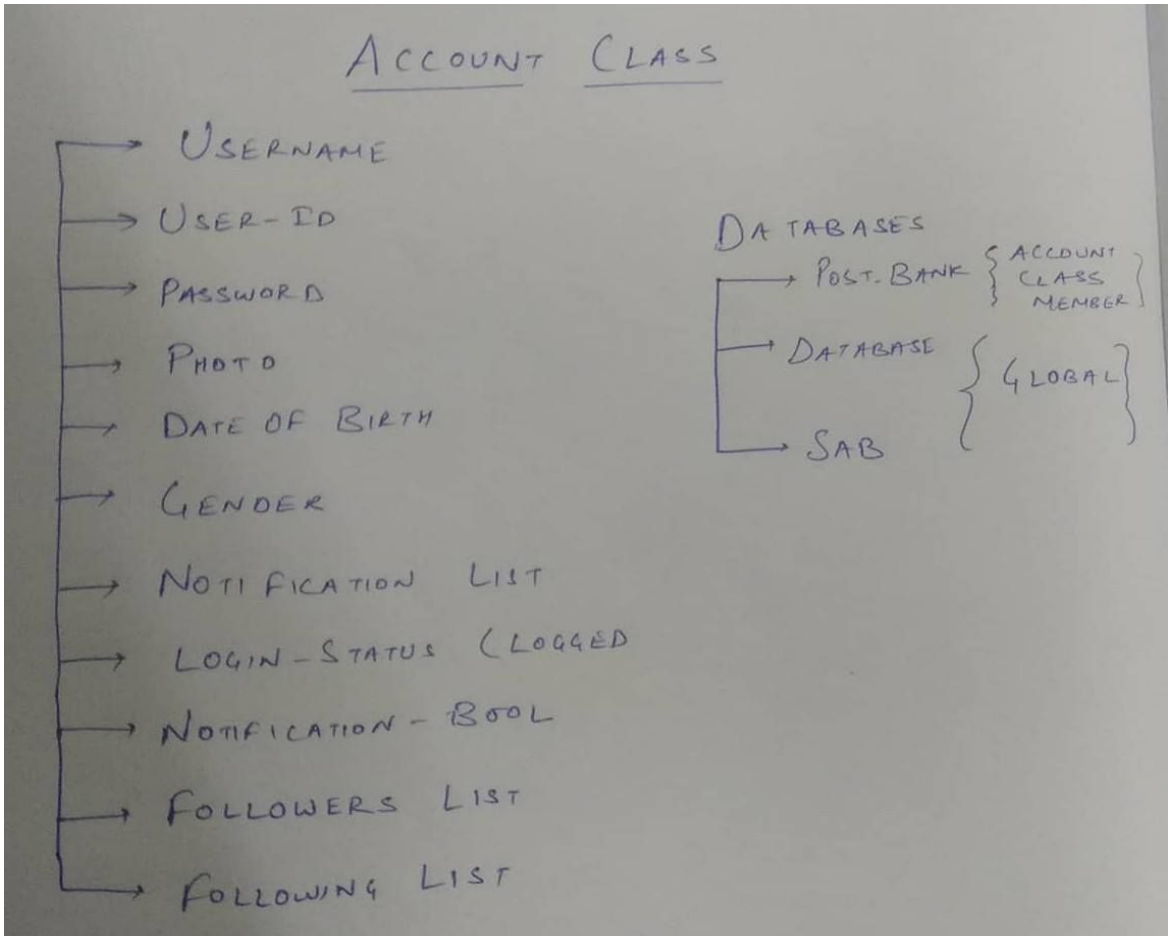
Description :

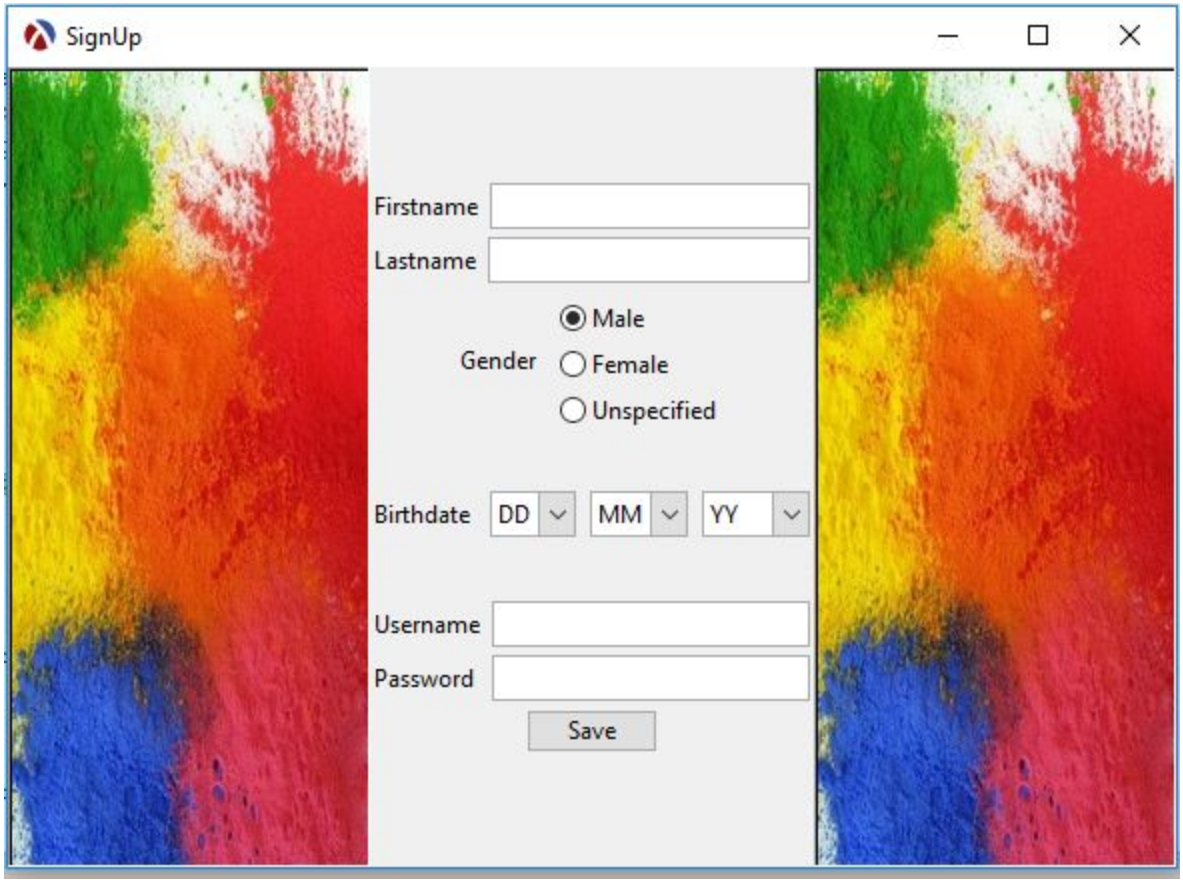
In this fast-growing world, networking is an inescapable aspect.
We here try to implement a basic version of twitter handle that you generally see. It will include Post, Replies to Post, Search, Notification bar, Following and Follower features.
A person can put posts on his timeline and even reply to other people posts.
Notification bar will ensure that the user is well informed about the latest replies on his post.
Search option provides us with list of all the users, amongst which the first user can decide upon whom to follow.
Follow option enables one user to view posts of other user.

Basic Design of Code

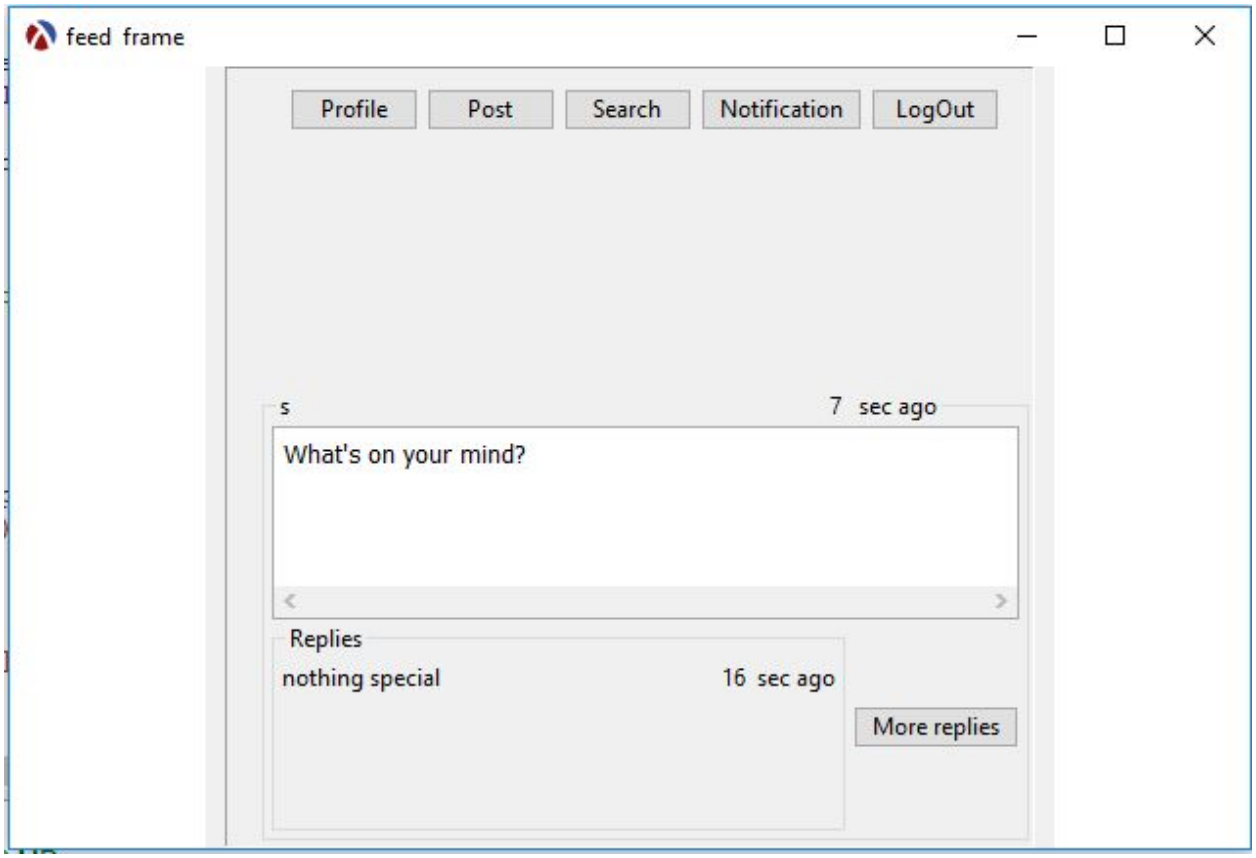
Core Foundation:

At the base level, we have an account class which takes basic information about the user.
This user class has inbuilt functions for making new post, new replies, etc.
Post and Replies are stored as structs as they include various attributes of them like content, post-ids, time of post, etc.
We have a Notification bar which gets updated each a reply to a post is made.


ACCOUNT CLASS
USERNAME
USER-ID
PASSWORD
PHOTO
DATE OF BIRTH
GENDER
NOTIFICATION LIST
LOGIN-STATUS (LOGGED
NOTIFICATION-BOOL
FOLLOWERS LIST
FOLLOWING LIST
DATABASES
POST-BANK
ACCOUNT CLASS MEMBER
DATABASE
SAB
GLOBAL


SignUp
Firstname
Lastname
Gender
Male
Female
Unspecified
Birthdate
DD
MM
YY
Username
Password
Save


intro
Welcome
it's cold outside....Enter!
create your own account
add posts and comments
share it to the world !
UserID
Password
Login
SignUp


feed frame
Profile
Post
Search
Notification
LogOut
s
7 sec ago
What's on your mind?
Replies
nothing special
16 sec ago
More replies

Databases :

1) **Post-bank** :- This is a hash contained inside account class. It maps post-id with post-struct. It is useful for printing feed.
2) **Database** :- This is a global hash which maps user-id with username and is useful for implementing the search option.
3) **Sab** :- This is the most important global hash mapping user-ids with the account objects. This hash helps us to achieve the job of interlinking various accounts.

GUI:

We have used following object% classes:

- **Frame%**
- **Panel%** (Vertical-panel% & Horizontal-panel%)
- **Field%**
- **Button%**
- **Canvas%**
- **Brush%**
- **Box** classes (Dialogue, Group, Radio)

SAMPLE I/O

INPUT :- We take inputs at 3 stages :-

- Input at the SignUp page -
- Input at the Login Page -
- This last input is taken in form of string which serves as the content for the post and replies.

OUTPUT :- Output will be a FEED containing all the post a particular person has made along with the posts of the person he has followed.

Limitations and Bug

- Due to limited screen space, we have kept an upper cap on number of posts displayed to be 20.
- Server Part :- we couldn't implement the server because :-
    1) Server sends and receives S-expressions only and we were dealing in objects.
    2) Racket documentation were not much clear about servers.

- Racket GUI has poor inbuilt mechanism for positioning of buttons. So, we needed to use hit and trial method to get the right button position.
- Frame size :- Canvas occupies whole of the area container space while we wanted it to occupy only a limited space
- we cannot directly access the replies from Notifications bar itself as that would lead to privacy breach in our account class.

Other Point of Interest :-

1) We have made the profile part as innovative as possible by adding images.
2) Also we have tried to make the interface look decent by adding background images.

Clever Coding :

1) Abstractions Used :-

i) **Dispatch function** - Similar to the bank account class, dispatch function here is used to call various functions. It helps us to maintain privacy of the user class as we can access it only using password of the user account.

ii) **Error-handling function** (alias DD) - This higher order function takes a text-string as an input and creates a pop-up box showing this text.

iii) **Printing functions** - In GUI in each frames there are functions which take raw form of data (struct, list, hash) and displays them in required form (group panel, message e.t.c.).

iv) **Clear field** - It clears the field in any frame.

v) **Get function** - Due to privacy issues, as other we cannot access elements of the account class from outside, we have made Get functions which returns a copy of that list, thus not enabling the person to alter the list from outside.

2) **Custom-size of panel%** - In Canvas% class it displays the whole canvas to the full area available, if we merge it with panel% then they would divide the frame into equal halves. To implement panel size we set some custom min width in that way the panel% got reduced.