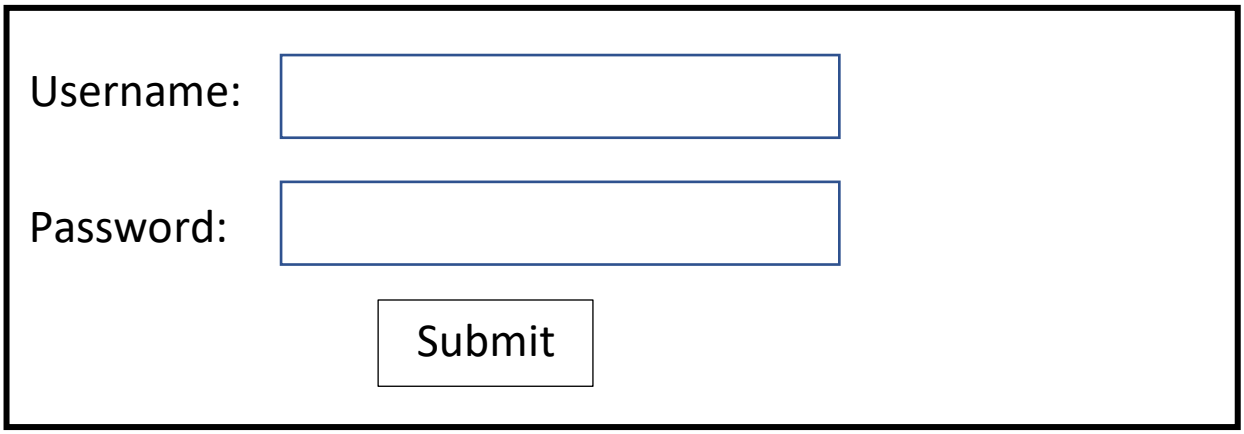
Collective Webapp

SRS

Introduction:
The Collective web application will enable the sharing of encrypted text among users of the system. All text will be encrypted and inaccessible for the admin. A user can enter and store text associated with their username into the database. This text is only unencrypted for the user who owns it, but the user who owns it can also share access to the unencrypted text to other users by their username. The owner can also remove the access granted to other users. In this way, we have a secure information sharing platform. I imagine writers, journalists, doctors, and scientists using the Collective web app to share their ideas and works in progress with their peers. In return the Collective web app would ensure the security of user's data. The users own their data, the Collective provides the service of keeping that data private, even from Collective itself.

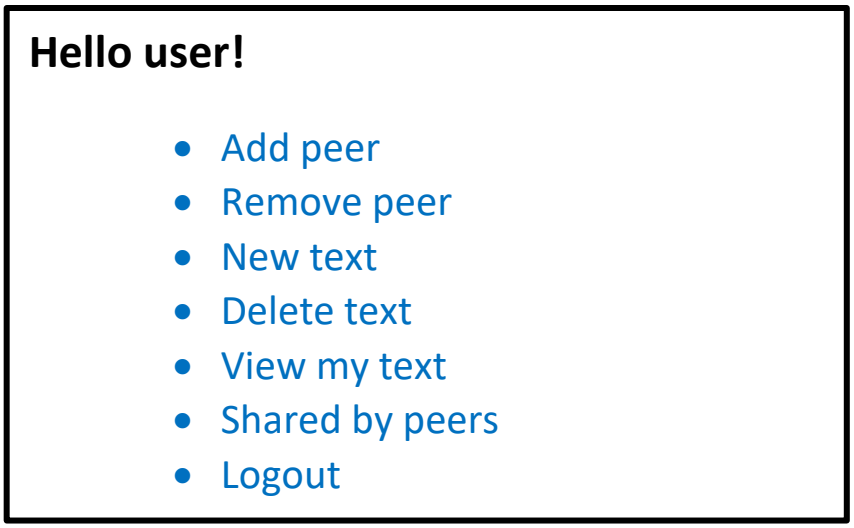
Overall Description:
The homepage is a login screen. All users must be added by the admin. When the admin logs in they will be have the options to 'add users', 'delete users', view 'logins', and add IPs to the 'whitelist'. When a user logs in they will have the options to 'add peer', 'remove peer', 'new text', 'delete text', 'view my text', 'shared by peers' and 'Logout'. All of these options will be hyperlinks that will update a post variable for switch statements.

For example, consider a user logging in:
Login



If the user is authenticated, then they will come to their main control page as depicted below.

User Control Page



From here the user can allow another user to view their encrypted text by going to 'Add peer'. Or the user can view what text they currently have stored encrypted in the database by clicking on 'View my text'. The user can click 'Shared by peers' to see a list of their peers (other users that this user has added as a peer) who have shared access to their unencrypted text. The usernames of the peers in the list of 'Shared by peers' are hyperlinks that will take the user to the 'View my text' of their peer who had shared it.

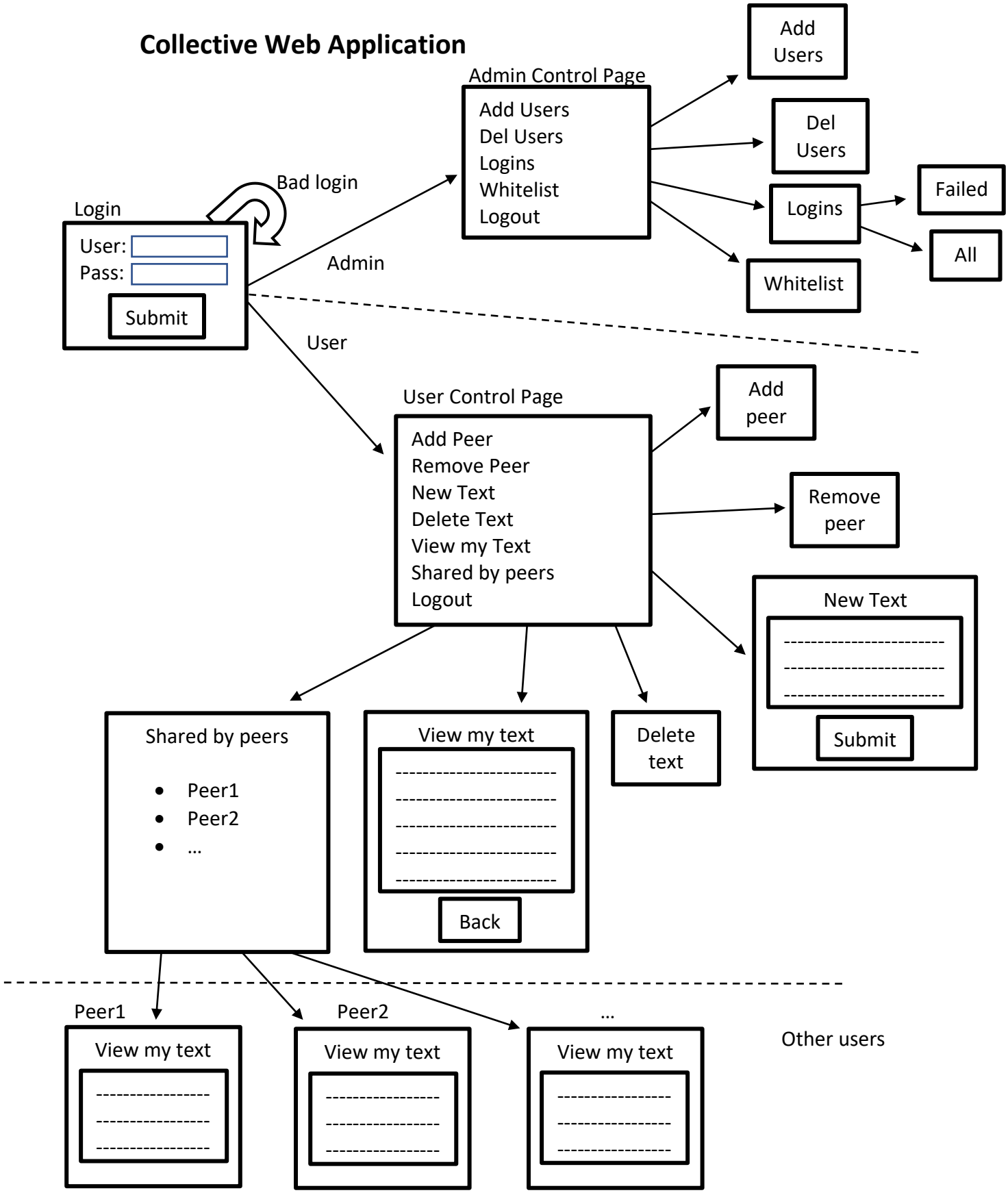
Specific Requirements:

Everything must be secured in layers. All input and output need to be sanitized and post variables need to be checked. User data must be encrypted in transit and at rest.

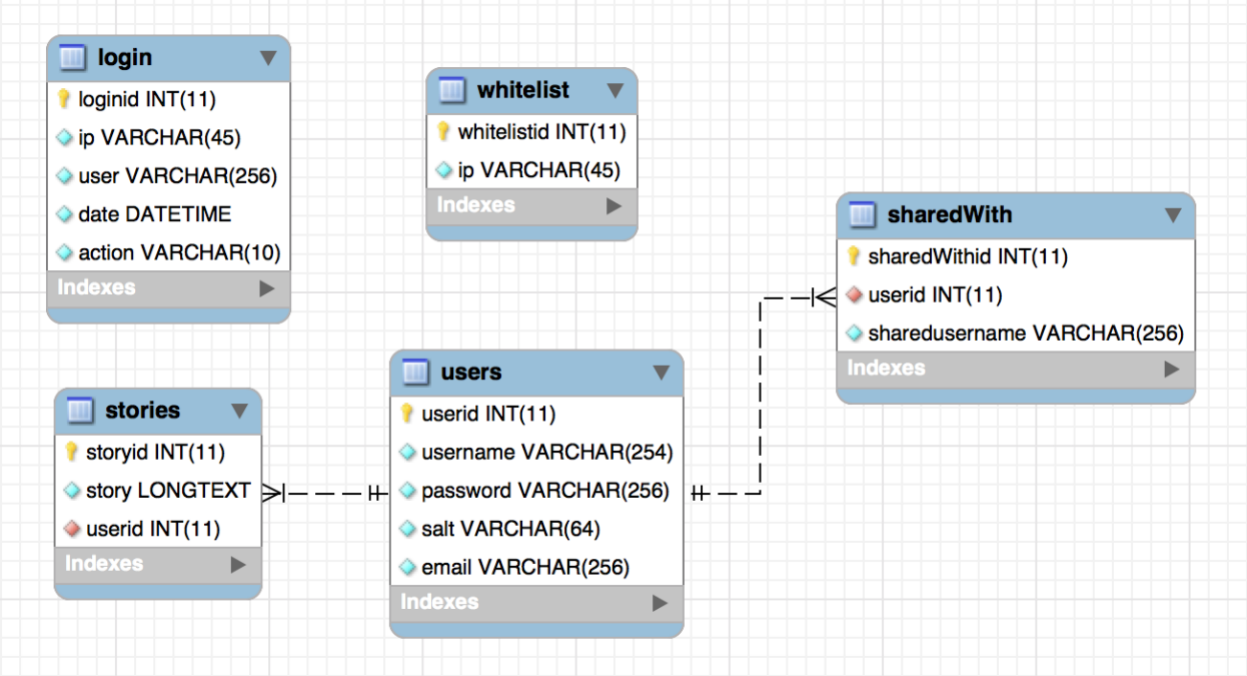
- Require authentication
- Logging all logins
- Watch failed logins
- Brute force password protection
- Securing webserver and database
- Firewall/iptables
- Redirect users to HTTPS
- Encrypt with SHA256 and a salt
- Use Sessions
- Use prepared statements
- Check for sqli
- Secure the application from the admin who may be a zealous dictator

The software stack will be php, html, bootstrap, mysql, apache, and mariadb. The webserver and database will be run on Fedora 26.

Supporting Information:



Database:


login
loginid INT(11)
ip VARCHAR(45)
user VARCHAR(256)
date DATETIME
action VARCHAR(10)
Indexes
whitelist
whitelistid INT(11)
ip VARCHAR(45)
Indexes
sharedWith
sharedWithid INT(11)
userid INT(11)
sharedusername VARCHAR(256)
Indexes
stories
storyid INT(11)
story LONGTEXT
userid INT(11)
Indexes
users
userid INT(11)
username VARCHAR(254)
password VARCHAR(256)
salt VARCHAR(64)
email VARCHAR(256)
Indexes