Bilkent University

Department of Computer Engineering

Database Systems Project

Project: Collaborative Hypertext Dictionary

Weblink: https://github.com/gunduzhuseyn/CS353Bilkent

Project Design

Section 1: Gunduz Huseynli, Muhammad Hamza Khan, Xheni Caka

Section 2: Mehmet Furkan Dogan

November 20, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Project of course CS353

Contents

ER Diagram

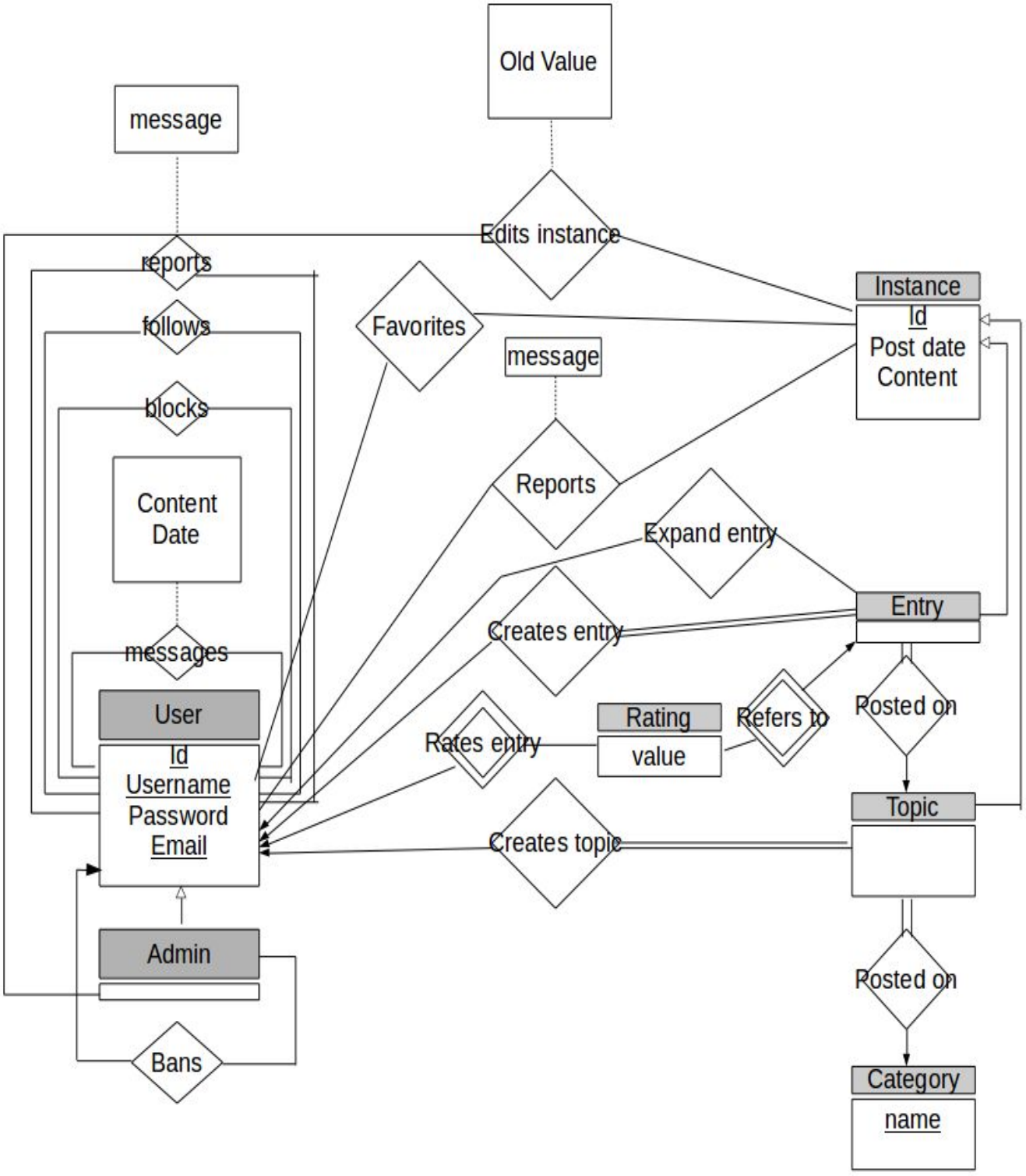

Old Value
message
Edits instance
reports
follows
blocks
Favorites
message
Instance
Id
Post date
Content
Content
Date
Reports
Expand entry
messages
Creates entry
Entry
User
Id
Username
Password
Email
Rates entry
Rating
value
Refers to
Posted on
Topic
Creates topic
Admin
Bans
Posted on
Category
name

Table Schemas

User(ID, username, email, password, isAdmin, isBanned)
UserMessage(messageContent, messageDate, senderID, receiverID)
UserBlock(blockerID, blockedID)
UserFollow(followerID, followedID)
AdminBan(adminID, bannedID, date)
AdminHistory(adminID, instanceID, oldValue, date, isInstanceTopic)
Report(userID, message, reportedID, date, reportType)
Entry(ID, date, content, userID, topicsID, rating)
Topic(ID, date, content, userID, categoryName)
Category(name)
Rating(userID, entryID, value)
Favorite(userID, instanceID, isInstanceTopic)

User Table Definition

```
CREATE TABLE `Servo`.`User`
        ( `ID` INT(20) NOT NULL AUTO_INCREMENT ,
        `username` VARCHAR(6) NOT NULL ,
        `email` VARCHAR(30) NOT NULL ,
        `password` VARCHAR(10) NOT NULL ,
        `isAdmin` BOOLEAN NOT NULL DEFAULT FALSE ,
        `isBanned` BOOLEAN NOT NULL DEFAULT FALSE ,
        PRIMARY KEY (`ID`), UNIQUE (`username`), UNIQUE (`email`));
```

UserMessage Table Definition

```
CREATE TABLE `Servo`.`UserMessage`
        ( `messageContent` VARCHAR(140) NOT NULL ,
        `messageDate` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ,
        `senderID` INT(20) NOT NULL ,
        `receiverID` INT(20) NOT NULL,
        FOREIGN KEY (`receiverID`) REFERENCES `User`(`ID`)
                ON DELETE CASCADE,
        FOREIGN KEY (`senderID`) REFERENCES `User`(`ID`)
                ON DELETE CASCADE) ;
```

UserBlock Table Definition

```
CREATE TABLE `Servo`.`UserBlock`(
```

```
    `blockerID` INT(20) NOT NULL ,
    `blockedID` INT(20) NOT NULL,
    FOREIGN KEY (`blockerID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE,
    FOREIGN KEY (`blockedID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE) ;
```

UserFollow Table Definition

```
CREATE TABLE `Servo`.`UserFollow`(
    `followerID` INT(20) NOT NULL ,
    `followedID` INT(20) NOT NULL,
    FOREIGN KEY (`followerID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE,
    FOREIGN KEY (`followedID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE) ;
```

AdminBan Table Definition

```
CREATE TABLE `Servo`.`AdminBan`(
    `adminID` INT(20) NOT NULL ,
    `bannedID` INT(20) NOT NULL,
    `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ,
    FOREIGN KEY (`adminID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE,
    FOREIGN KEY (`bannedID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE) ;
```

AdminHistory Table Definition

```
CREATE TABLE `Servo`.`AdminHistory` (
    `adminID` INT(20) NOT NULL ,
    `instanceID` INT(20) NOT NULL ,
    `oldValue` VARCHAR(140) NOT NULL ,
    `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ,
    `isInstanceTopic` BOOLEAN NOT NULL,
    FOREIGN KEY (`adminID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE);
```

Reports Table Definition

```
CREATE TABLE `Servo`.`Report` (
    `userID` INT(20) NOT NULL ,
    `reportedID` INT(20) NOT NULL ,
```

```
    `message` VARCHAR(140) NOT NULL ,
    `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ,
    `reportType` INT(2) NOT NULL,
    FOREIGN KEY (`userID`) REFERENCES `User`(`ID`)
        ON DELETE CASCADE);
```

Category Table Definition

```
CREATE TABLE `Servo`.`Category` (
    `name` VARCHAR(20) NOT NULL ,
    PRIMARY KEY (`name`));
```

Topic Table Definition

```
CREATE TABLE `Servo`.`Topic` (
    `ID` INT(20) NOT NULL AUTO_INCREMENT ,
    `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ,
    `content` VARCHAR(140) NOT NULL ,
    `userID` INT(20) NOT NULL ,
    `categoryName` VARCHAR(20) NOT NULL ,
    PRIMARY KEY (`ID`),
    FOREIGN KEY (`userID`) REFERENCES `User`(`ID`),
    FOREIGN KEY (`categoryName`) REFERENCES `Category`(`name`) );
```

Entry Table Definition

```
CREATE TABLE `Servo`.`Entry` (
    `ID` INT(20) NOT NULL AUTO_INCREMENT ,
    `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP ,
    `content` VARCHAR(140) NOT NULL ,
    `userID` INT(20) NOT NULL ,
    `topicsID` INT(20) NOT NULL ,
    `rating` INT(20) NOT NULL DEFAULT 0,
    PRIMARY KEY (`ID`),
    FOREIGN KEY (`userID`) REFERENCES `User`(`ID`),
    FOREIGN KEY (`topicsID`) REFERENCES `Topic`(`ID`) );
```

Rating Table Definition

```
CREATE TABLE `Servo`.`Rating` (
    `userID` INT(20) NOT NULL ,
    `entryID` INT(20) NOT NULL ,
    `value` INT(2) NOT NULL ,
    PRIMARY KEY (`userID`, `entryID`),
```

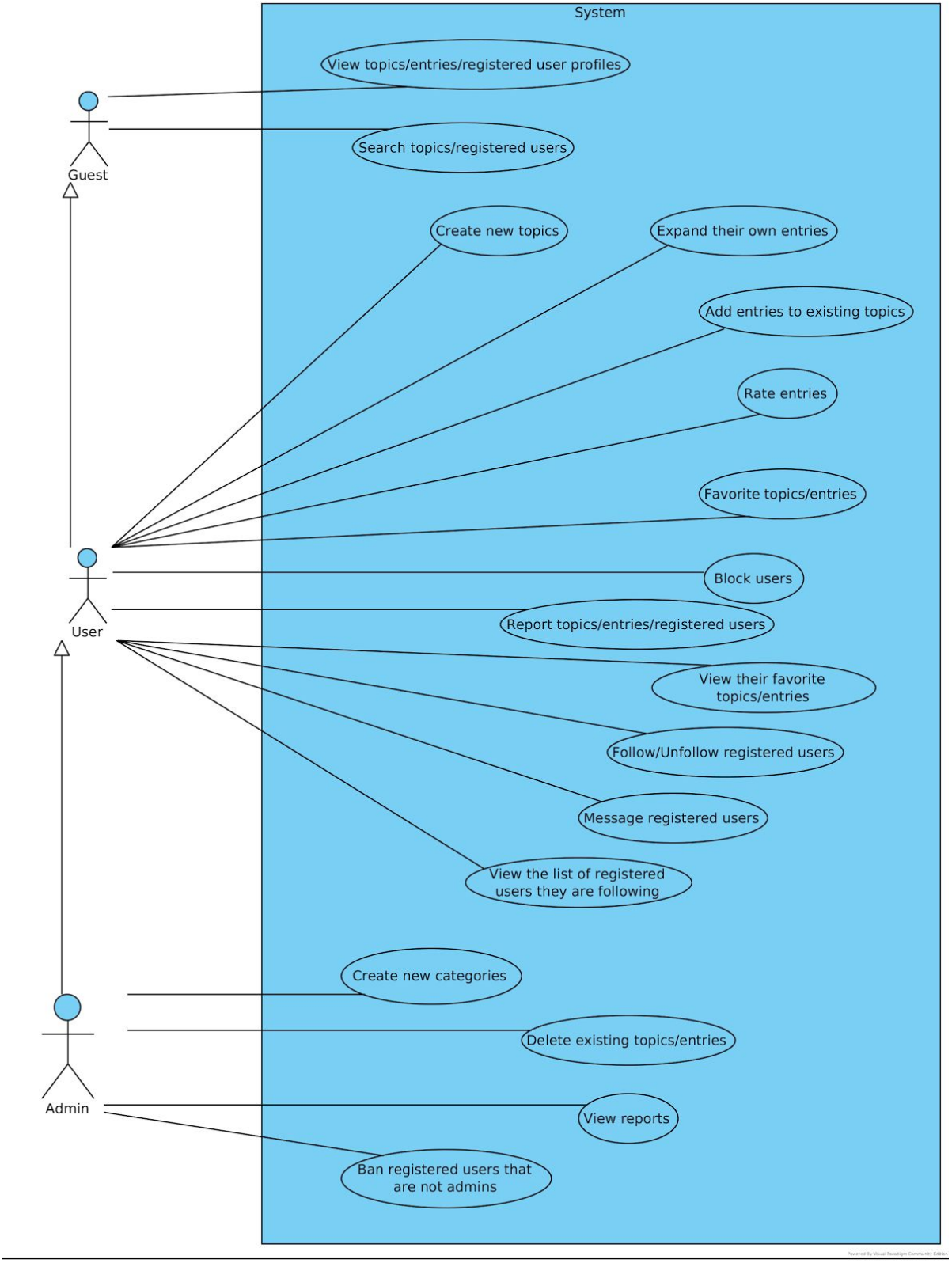
```
    FOREIGN KEY (`userID`) REFERENCES `User`(`ID`),
    FOREIGN KEY (`entryID`) REFERENCES `Entry`(`ID`) );
```

Favorite Table Definition

```
CREATE TABLE `Servo`.`Favorite` (
    `userID` INT(20) NOT NULL ,
    `contentID` INT(20) NOT NULL ,
    `isInstanceTopic` BOOLEAN NOT NULL ,
    PRIMARY KEY (`userID`, `contentID`, `isInstanceTopic`),
    FOREIGN KEY (`userID`) REFERENCES `User`(`ID`));
```

Functional Components

Use Cases:



Use Case Diagram

There are three main groups of users for our platform. Users who are not registered or logged in (Guest), users who are registered and are logged in (User), and administrators of the platform (Admin).

Guests can search topics, entries, or users and view them without logging in to the platform. They need to register and log in to the platform to do more. Users, who are logged in to the platform, can do all the actions that Guest users can do. Additionally, they can create new topics, expand currently available topics with new entries, and change their own entries. Users can favorite some entries and topics to access them more easily. Moreover, they can rate the entries with -1, or +1 points to alter the popularity of that entry. Also they can interact with other users, by following them to reach their content easily. They can message each other to communicate privately and easily. Also, users can unfollow or block each other, if they do not get along with others. Finally, they can report the inappropriate content for the admins to review.
Admins will be able to review the reports and delete content that they think should not be on the web page. Additionally they can ban the users that are repeatedly breaking the platform's rules and misbehaving. Admins will also be able to create new categories for topics.

Data Structures and Implementation Algorithms

For the implementation we are using common sql data structures such as int and varchar. We are going to store the the entries in an descending order according to their ranks.

Searching: Users will provide keywords to the search bar, and relevant users, topics, and entries will be shown as results.

Authentication: Hashes of the users' passwords will be saved in the platform for enhanced security. Also for password resetting a random generated string will be sent to user's registered e-mail, and will be used to authorize password reset.

Proper security measures will be taken to prevent malicious access to the platform.

UI Design and SQL queries

GUI Design

Login Page

This will be the landing page of our website for anyone who visits www.servo.com. Visitors will be able to:

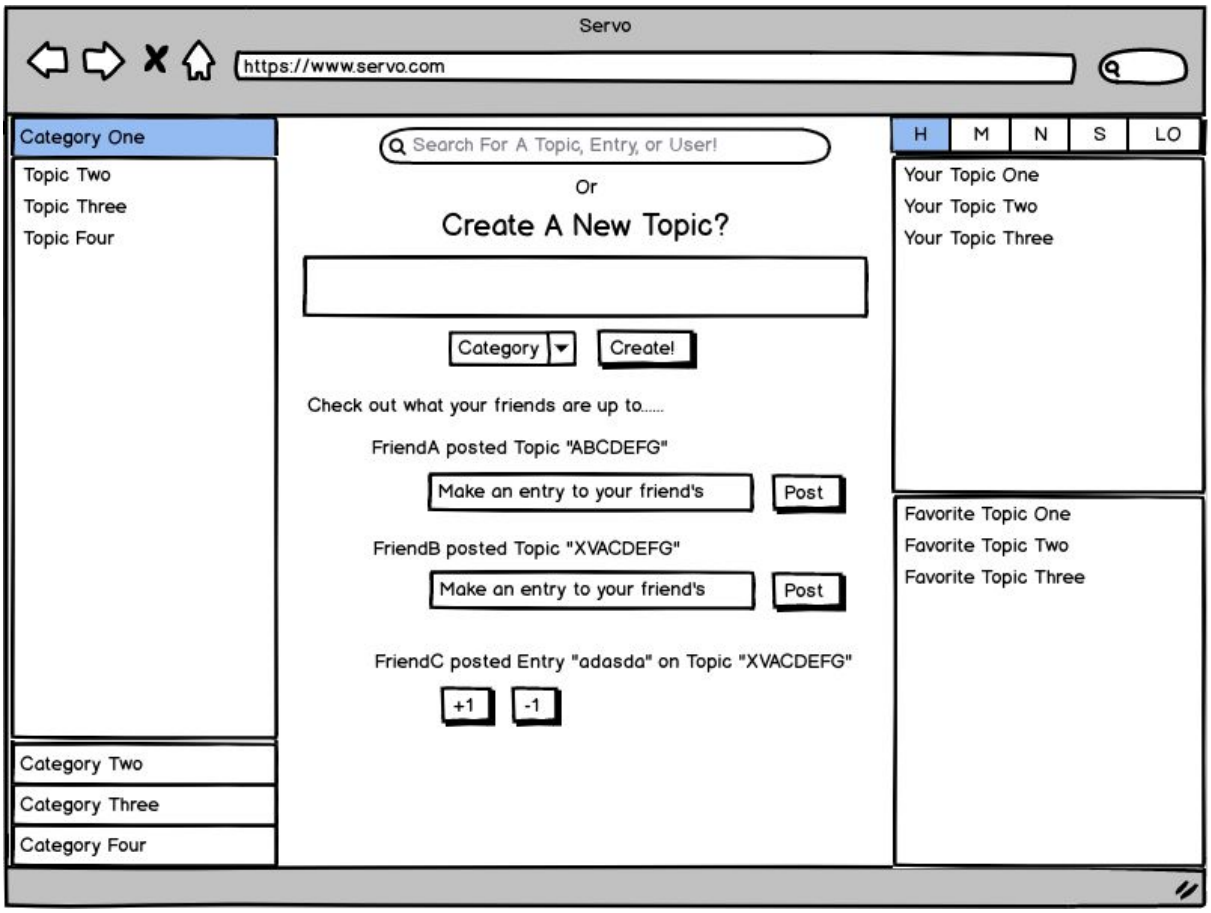
- Login into their accounts from here by either typing in their username or email along with their password. And then clicking on the “Lets Go!” button.
- Reset their passwords by either providing their username or email and then clicking on “Reset Password?” button.

- Sign up to the website by clicking on the “Create A New One…” button. This action will redirect the visitors to the sign up page.
- Clicking on the “Lets Go!” button without providing anything will allow the visitor to explore the website as a guest. Guests will be able to read everything on the stuff but won’t be able to modify any of it.

Sign Up Page

Using this page visitors will be able to sign up with Servo. Users will be required to provide a unique user name and email address along with a password which must of at least 6 characters. Then the visitors would have to press the “Lets Go!” button for the sign up process to be completed.

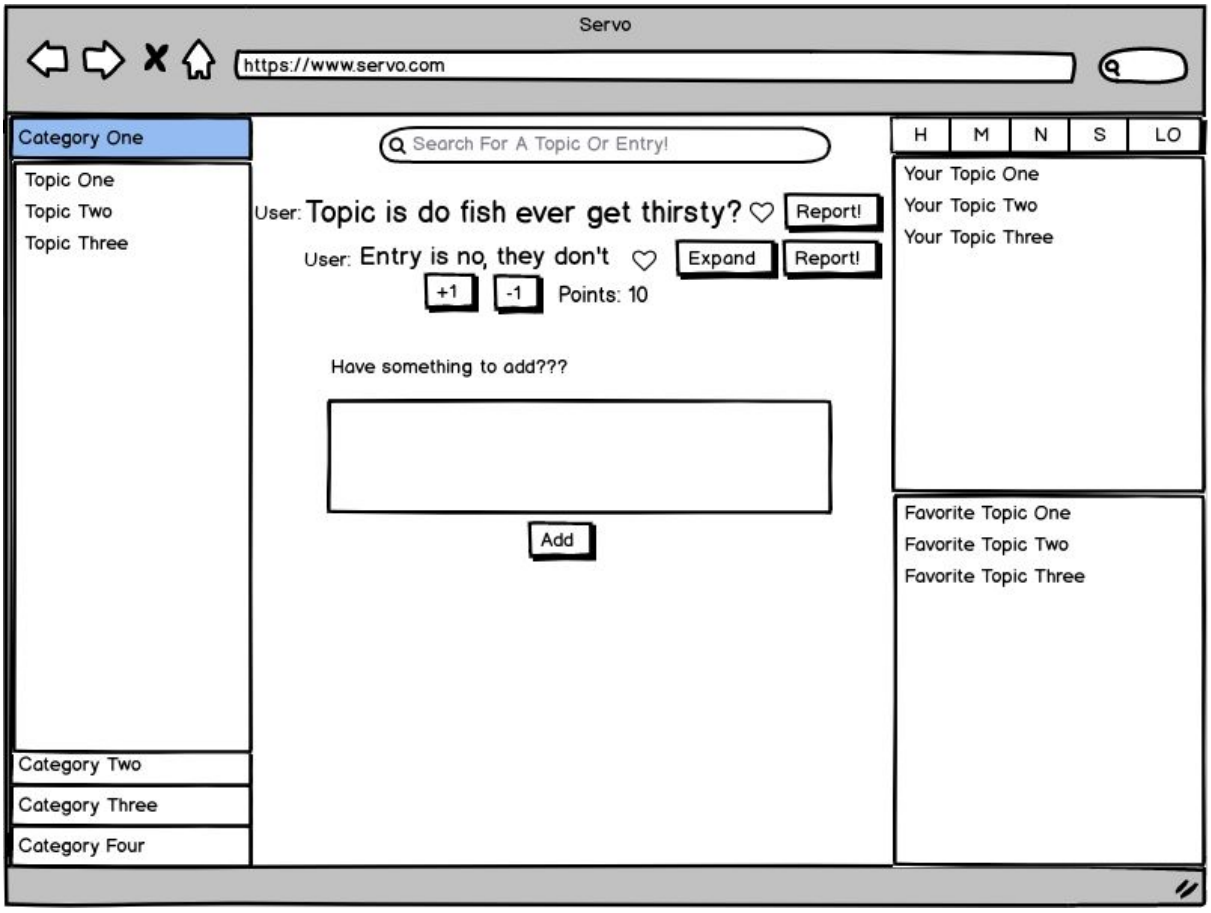
Home Page



Once the login/signup process is completed, the users will be redirected to this Home page of the website. On this page users will be able to:

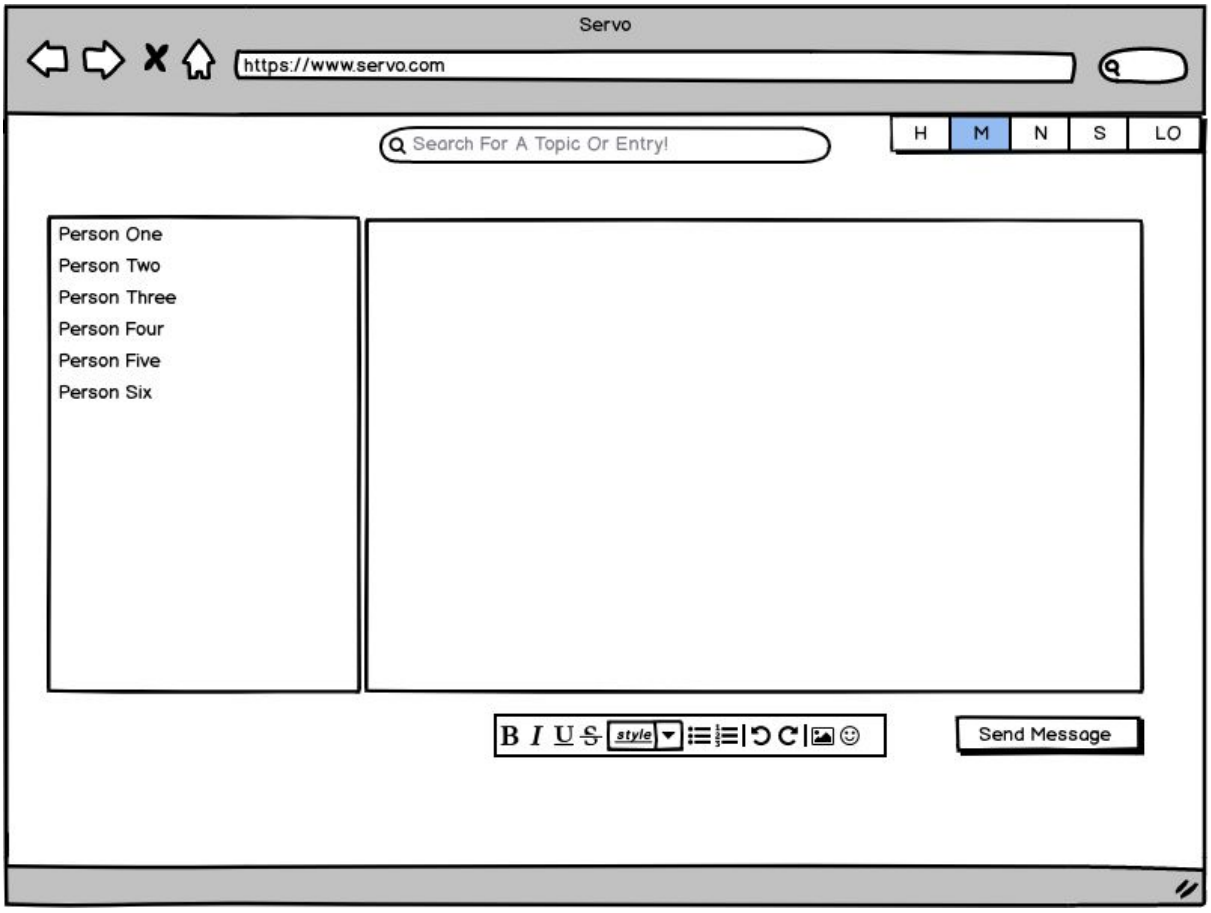
- Browse through different categories and topics under those categories, using the explorer on the left side of the page.
- Search for a specific Topic, Entry, or other Users using the search bar in the middle-top of the page.
- Create a new topic, under a specific category.
- The middle of the page will act like a feed showing topics and entries posted by people the user is following.
- The top right corner of the page will have buttons to redirect user to other parts of the website. The “H” button is for the “Home Page”, “M” for “Messages”, “N” for “Notifications”, “S” for “Settings”, and “LO” for “Logging Out” of the website.
- Below these buttons, the list of topics posted by the user will be displayed in chronological order.
- And on the bottom right corner of the page, the list of favorite topics of the user will be posted, again in chronological order.

Topic Page



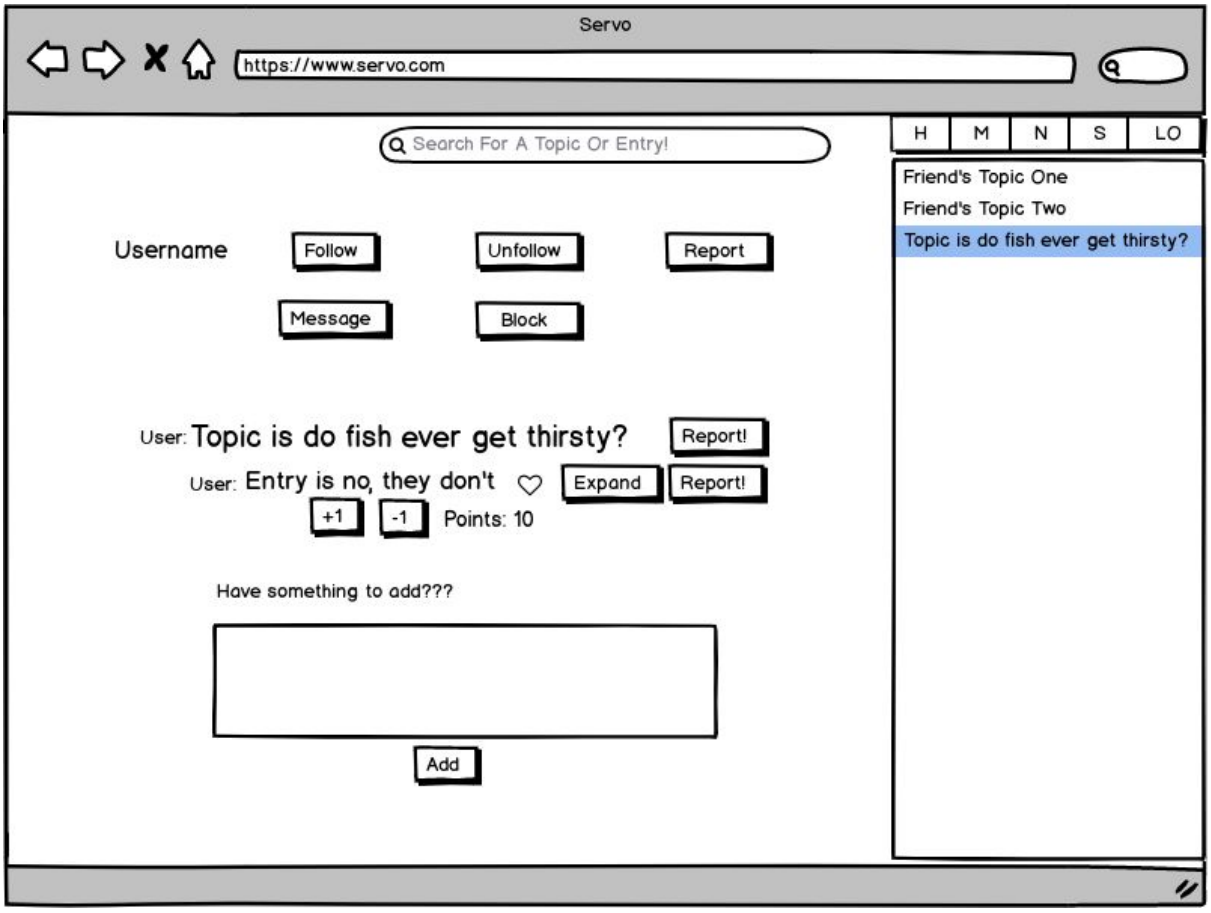
This page will be similar to the Home Page but with one key difference. The middle column of the page will include the Topic under discussion. On the top of this column, the name of the topic will be written, along with the buttons to report or favoritize it. Below that, there will be entries posted on the topic. The entries will have points associated with it. Each user can either add 1 to the points or subtract 1 from them. An option to add a new entry will also be displayed. A user may add as many entries as many he/she wants to. However, previous entries or topics won't be deleted. New content might be added to them by clicking on the Expand button. The usernames of the users who post the topics/entries will also be displayed.

Messages



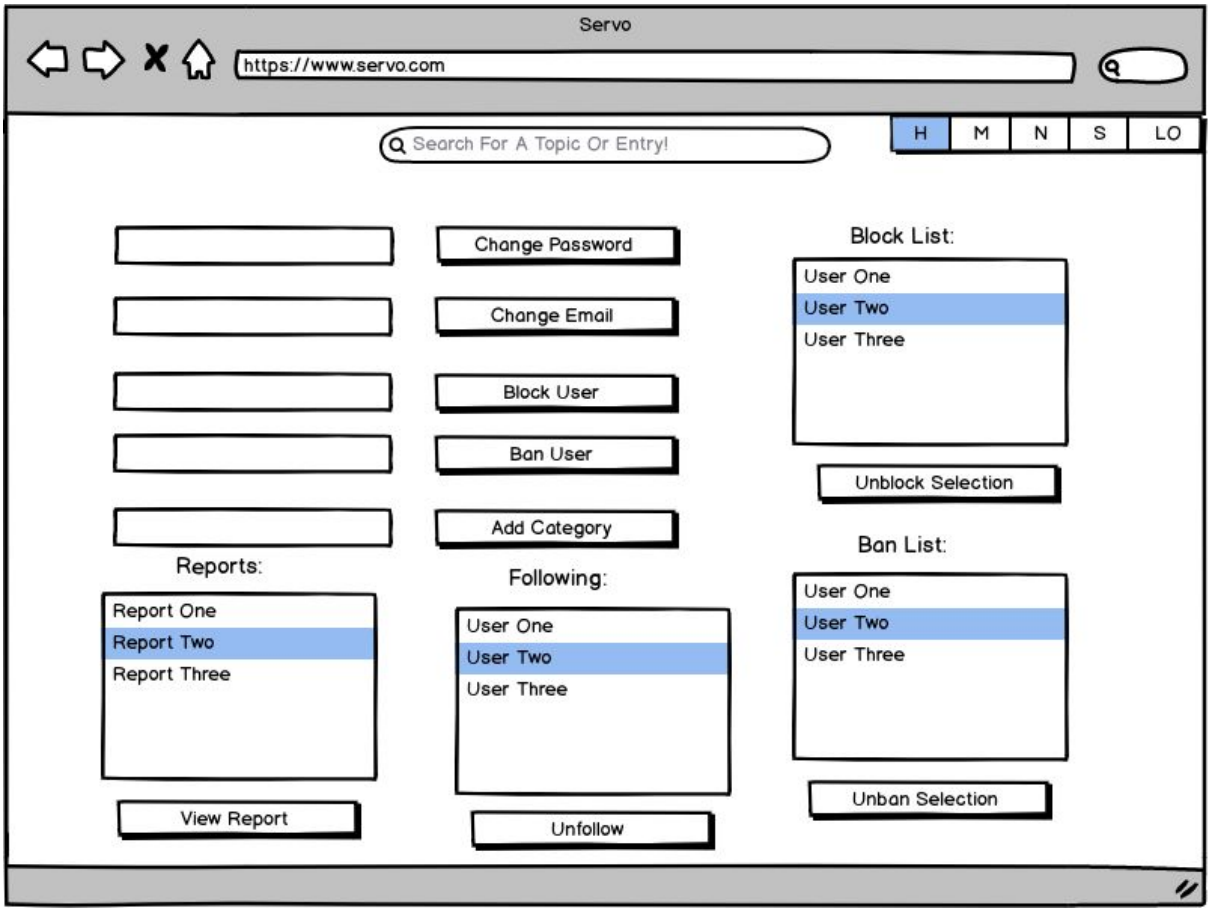
The message page will allow users to chat with other users. On the left of the page, there will be names of users with whom the user had previously interacted with. The user can either open up a previous chat or go to the profile of another user to start a new chat with that user.

Another User's Profile



Users will be able to display profiles of other users by either clicking on their names at any place in the website or by searching for them via the search bar. The profiles will include the usernames of the users along with the buttons to either Follow, Unfollow, Report, Message, or Block them. On the right side of the page, the topics posted by that user will also be displayed. A topic can be selected from that list to be interacted with. This interaction is similar to the one on the Discussion Page.

Settings of an Admin



The settings pages will be different for Admins and Users. Admin's' settings page will look like the above one. From this page, an Admin will be able to:

- Change his personal password or email.
- Block other users from messaging or following him.
- Ban other users from using the site.
- Add a new category to the website.
- View reports submitted by other users.
- Look at the list of users, he is following, and be able to unfollow any of them.
- Look at the list of users which are banned from the website and be able to unban any of them.
- View all of the users blocked by the admin and be able to unblock any of them.

Settings of a user

From the Settings page of the normal user, the user will be able to:

- Change his/her password or email.
- Block any user by typing his username.
- View previously Blocked users and unblock any of them.
- View the list of people he/she is following and be able to unfollow any of them.

SQL Codes For Operations

Create User
$username, $email, $password
INSERT INTO User (username, email, password) VALUES ($username, $email, $password)

Create Topic
$content, $topicTag, $userID
INSERT INTO Topic (content,receiverID, senderID) VALUES (SELECT)
VALUES($content, $topicTag, $userID)

Create Entry
$content, $topicID, $userID
INSERT INTO Entry (content, topicID, userID)

VALUES($content, $topicID, $userID)

Expand Entry
$entryID, $expandedContent
UPDATE Entry SET (content = CONCAT(content, $expandedContent)) WHERE Entry.ID == entryID

Rate Entry
$rating, $userID, $entryID
INSERT INTO Rating (userID, entryID, value) VALUES ($userID, $entryID, $rating);

Report User/Topic/Entry
$reportedID, $reportType, $userID, $message
INSERT INTO Report (reportedID, reportType, userID, message)
VALUES ($reportedID, $reportType, $userID, $message)

Follow User
$followerID, $followedID
INSERT INTO UserFollow (followerID, followedID) VALUES ($followerID, $followedID)

Block User
$blockerID, $blockedID
INSERT INTO UserFollow (blockerID, blockedID) VALUES ($blockerID, $blockedID)

Favorite Topic
$userID, $instanceID
INSERT INTO Favorite (userID, instanceID, isInstanceTopic) VALUES ($blockerID, $blockedID, 1)

Favorite Entry
$userID, $instanceID
INSERT INTO Favorite (userID, instanceID, isInstanceTopic) VALUES ($userID, $instanceID, 0)

Show Favorite Entries
$userID
SELECT * FROM Entry_Combined_View WHERE ID in (SELECT Favorite.contentID FROM Favorite WHERE userID = $userID && isInstanceTopic = 0) ORDER BY date DESC

Show Favorite Topics
$userID
SELECT * FROM Topic_Combined_View WHERE ID in (SELECT Favorite.contentID FROM Favorite WHERE userID = $userID && isInstanceTopic = 1) ORDER BY date DESC

View followers
$userID

SELECT username, ID FROM users JOIN UserFollow ON UserFollow.followerID = User.ID
WHERE $userID = UserFollow.followedID

View following
$userID
SELECT username, ID FROM users JOIN UserFollow ON UserFollow.followedID = User.ID
WHERE $userID = UserFollow.followerID

Unfollow A User
$userID, $followedID
DELETE * FROM UserFollow WHERE (UserFollow.followerID = $userID &&
UserFollow.followedID = $followedID)

View A User
$userID
SELECT * FROM Entry_Combined_View AS E WHERE E.userID = $userID ORDER BY
date DESC

Search User
$username
SELECT ID, username FROM User WHERE username LIKE (“%$username%”)

Search Topic
$keyword
SELECT ID, content, username FROM User WHERE username LIKE (“%$keyword%”)
ORDER BY date

Create New Categories
$newCategory
INSERT INTO Category (name) VALUES ($newCategory)

Delete Topics
$topicID
DELETE * FROM Topic WHERE ID = $topicID

Delete Entries
$entryID
DELETE * FROM Entry WHERE ID = $entryID

Ban User
$toBanID, $adminID
INSERT INTO AdminBan(adminID, bannedID) VALUES($adminID, $toBanID)

View Reports
SELECT message, userID, reportedID FROM Report
Change Email

$userID, $newEmail
UPDATE Servo.User SET email = $newEmail WHERE User.ID = $userID

Send Message
$userID, $content, $receiverID
INSERT INTO UserMessage(messageContent, senderID, receiverID) VALUES ($content, $userID, (SELECT ID FROM User LEFT JOIN UserBlock ON User.ID = UserBlock.blockedID WHERE ID = $receiverID && UserBlock.blockedID = NULL))

Receive Messages
$userID
WITH IDList(contactID) AS
((SELECT receiverID FROM UserMessage WHERE senderID = $userID) UNION (SELECT senderID FROM UserMessage WHERE receiverID = $userID))
WITH contactList(contactName) AS (SELECT username FROM User WHERE User.ID in IDList)
SELECT contactID, contactName, messageContent, date FROM contactList, IDList, UserMessage WHERE (UserMessage.senderID = $userID && UserMessage.receiverID in IDList) || (UserMessage.receiverID = $userID && UserMessage.senderID in IDList)
GROUP BY contactName
ORDER BY date

Followed Peoples Entry History
$userID
WITH FollowedList(ID, UName) AS (SELECT ID, User.username AS UName FROM User JOIN UserFollow ON (UserFollow.followedID = User.ID && followerID = $userID))
SELECT * FROM Entry_Combined_View WHERE userID IN FollowedList
---
Followed Peoples Topic History
$userID
WITH FollowedList(ID, UName) AS (SELECT ID, User.username AS UName FROM User JOIN UserFollow ON (UserFollow.followedID = User.ID && followerID = $userID))
SELECT * FROM Topic_Combined_View WHERE userID IN FollowedList
---
Followed Peoples Rating History
$userID
WITH FollowedList(ID, UName) AS (SELECT ID, User.username AS UName FROM User JOIN UserFollow ON (UserFollow.followedID = User.ID && followerID = $userID))
SELECT * FROM Rating JOIN Entry_Combined_View ON Rating.entryID = Entry_Combined_View.ID WHERE Rating.userID IN FollowedList
---
Change Password
$userID, $newPw
UPDATE User SET (password = $newPw) WHERE $userID = ID

Advanced Database Components

Views

Entry Combined View
Combines, owners names, and entry rankings with entries.

```
CREATE VIEW `Servo`.`Entry_Combined_View` AS
        SELECT
                Entry.rating,
                Entry.content,
                Entry.ID,
                User.ID AS userID,
                User.username,
                Entry.date,
                Topic.ID AS topicsID,
                Topic.content AS topicName
        FROM User, Topic,Entry
        WHERE (Entry.userID = User.ID) && (Entry.topicsID = Topic.ID)
        GROUP BY Entry.ID
```

Topic Combined View
Combines topics with owner names.

```
CREATE VIEW `Servo`.`Topic_Combined_View` AS
        SELECT
                User.username,
                Topic.ID,
                Topic.content,
                Topic.categoryName,
                Topic.userID,
                Topic.date
        FROM User, Topic
        WHERE (Topic.userID = User.ID)
```

Triggers

The rating value should be updated when a user gives a new rating to an entry

```
CREATE trigger userrating after insert on Rating
referencing new row as newrow
FOR each row
WHEN exists (SELECT R.value
            FROM Rating R
            WHERE newrow.userId = R.userID )
BEGIN atomic
      update Entry
      set rating = (select rating
                  from Rating R
                  group by R.entryID)
      where Entry.ID = newrow.ID;
end;
```

Constraints

- Password cannot be shorter than 6 characters
- A user can rate an entry only once
- A user cannot delete an entry
- Usernames are unique
- User's email is unique
- Reading the entries does not require login.
- Rating and adding a new entry can be done only if logged on
- Adding an image is not allowed