CS353-Database Systems

Project Design Report

Social Discussion Website

Section 1 / Group 9

Furkan Salih Taşkale - 21300878

Mustafa Culban - 21301187

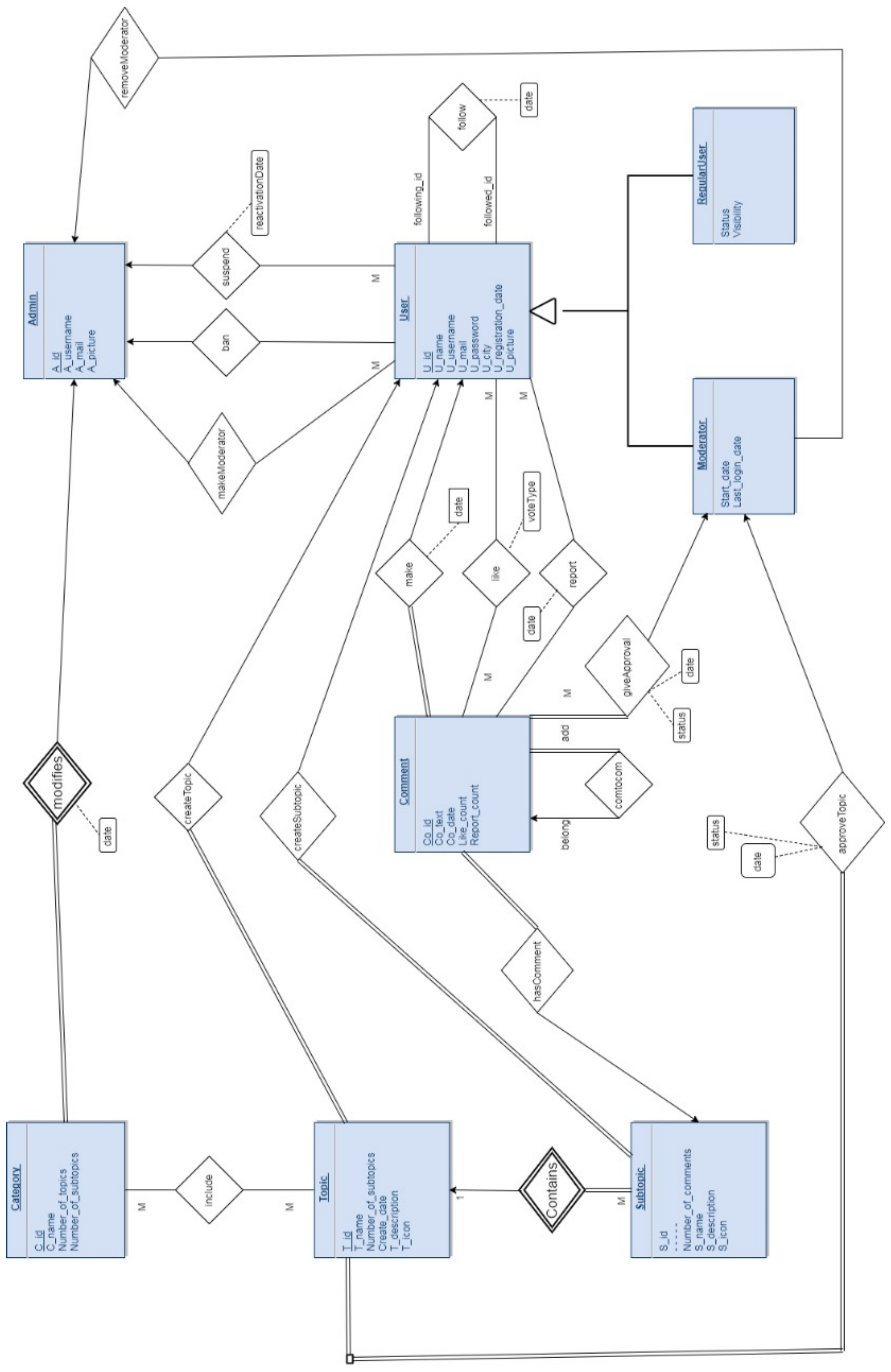
İzel Gürbüz - 21301018

Aldo Tali - 21500097

Due Date – 20th of November 2017

Table of Contents

1. Revised E/R Model



We revised our E/R model based on the feedback given in the Proposal of the project by the assistant as follows:

- The cardinality constraints for several of our relations were changed to specify more meaning to the semantics of our application. The relation modifies between Admin and Category entities were made a one to many relationship indicating that an admin can modify many categories. Similarly the relation make between User and Comment entities and the relation approve between Moderator and Topic were changed to a one to many for the same reasoning as the previous case. Total participation was added for one side of the recursive relation comtocom (Read as Comment to Comment) within the Comment Entity and a cardinality constraint was added to the other so that a reply to a comment will belong to another comment and all such comments will be made as replies to other comments.
- The number of weak entity sets was reduced in order to avoid redundant dependencies or ternary weak entities. For example the relations makes between User and Comment, HasComment between Comment and Subtopic and ApproveTopic between Moderator and Topic were changed from weak entity relations to normal relations with total participation. The total participation was kept to imply the complete involvement of one entity in the relation.
- Relations like between User and Comment entities was given the attribute votetype which should keep the information whether a user likes or dislikes a given comment. Similarly in the giveApproval relation between Moderator and Comment the attribute status was added to store the approve status information for each comment based on the evaluations the Moderator will make.

- Relations report, like and makes were adjusted to show relations between User and Comment entities instead of User and Comment Aggregation.

We have also made the following changes in our initial E/R diagram:

- Due to misuse in the proposal the aggregations used to semantically encapsulate the Comment entity and comtocom relation into one as well as the aggregation encapsulating the Topic and Subtopic entities were removed from the E/R model.
- The Favorite entity along with add, discard and embed relations were removed as they were considered to be redundant in the design on the application, since there is already a functionality to follow users based on which the comments of those users can be seen recursively defined in the User entity.
- The administrate relation between Admin and User was changed to four separate relations. The first one is ban and the second one is suspend. Both these relations continue to be defined as relations between Admin and User. The cardinality constrains for this case were set to be one to many. Lastly the Suspend relation will also keep an attribute for the reactivationDate which is the date when the suspension finishes. The other two are makeModerator a relation between Admin and User and removeModerator a relation between Admin and Moderator
- The create relation between User and Topic was also divided into two relations, one corresponding to the creation of the topic and the other corresponding to the creation of the subtopic. These relations are made with a total participation since all topics or subtopics, if they exist then they have been created by a user and a one to many cardinality mapping is used.

- We changed the name of moderates relation between Moderator and Comment entities to give Approval. Similarly we changed to name of approve into approveSubtopic.
- The isApproved attribute was removed from the Topic, Subtopic and Comment entities and given to the giveAprroval relationships between the entities. This was purely a design choice to make the database structure more meaningful

2. Relation Schemes

a. Category

Relational Model

Category(c_id, c_name, number_of_topics,)

Functional Dependencies

c_id-> c_name,number_of_topics

c_name-> c_id,number_of_topics

Candidate Keys

{(c_id), (c_name)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `Category` (
`c_id` INT(32) NOTNULL AUTO_INCREMENT ,
`c_name` VARCHAR(64) NOT NULL ,
`number_of_topics`INT NOT NULL ,
`number_of_subtopics` INT NOT NULL ,
PRIMARY KEY(`c_id`)) ENGINE = InnoDB,
UNIQUE (`c_name`);
```

b. Topic

Relational Model

Topic(t_id,t_name,number_of_subtopics,create_date,t_description,t_icon)

Functional Dependencies

t_id->t_name, number_of_subtopics, create_date,t_description,t_icon

t_name-> t_id, number_of_subtopics,create_date,t_description,t_icon

Candidate Keys

{(t_id),(t_name)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE  `Topic` ( `t_id` INT NOT NULLAUTO_INCREMENT ,
`t_name` VARCHAR(64) NOT NULL ,
`number_of_subtopics`INT NOT NULL ,
`create_date` VARCHAR(32) NOT NULL ,
`t_description`VARCHAR(255) NOT NULL ,
`t_icon` VARCHAR(255) NOT NULL ,
PRIMARY KEY(`t_id`)) ENGINE = InnoDB;
```

c. Subtopic

Relational Model

Subtopic(S_id,T_id, Number_of_comments, S_name, S_Icon, S_description)

Foreign key T_id references Topic

Functional Dependencies

s_id,t_id-> Number_of_comments,S_name,S_Icon,S_description

Candidate Keys

{(S_id, T_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `SubTopic` ( `S_id` INT NOT NULL , `T_id`INT NOT NULL
, `Number_of_comments` INT NOT NULL , `S_name` VARCHAR(255)
NOTNULL , `S_Icon` VARCHAR(255) NOT NULL , `S_description` TEXT
NOT NULL ,PRIMARY KEY (`S_id`, `T_id`),
FOREIGN KEY (`T_id`) REFERENCES `Topic`(`t_id`)
) ENGINE = InnoDB;
```

d. Comment

Relational Model

Comment(Co_id, Co_text, Co_date, Like_count, Report_Count, Dislike_Count)

Functional Dependencies

Co_id -> Co_text,Co_date,Like_count,Report_Count,Dislike_Count

Candidate Keys

{(Co_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `Comment` (
`Co_id` INT(64) NULL DEFAULTNULL AUTO_INCREMENT ,
`Co_text` TEXT NULL DEFAULT NULL ,
`Co_date` VARCHAR(32)NULL DEFAULT NULL ,
`Like_count` INT(8) NULL DEFAULT NULL ,
`Report_Count`INT(8) NULL DEFAULT NULL ,
`Dislike_Count` INT(8) NULL DEFAULT NULL ,

PRIMARYKEY (`Co_id`) ENGINE = InnoDB;
```

e. Admin

Relational Model

Admin(A_id, A_username, A_mail, A_picture, A_password)

Functional Dependencies

A_id->A_username,A_mail,A_picture,A_password

A_username -> A_id,A_mail,A_picture,A_password

A_mail->A_id,A_username,A_picture,A_password

A_password -> A_id,A_username,A_mail,A_picture

Candidate Keys

{(A_id),(A_username),(A_mail),(A_password)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `Admin` ( `A_id` INT NOT NULLAUTO_INCREMENT ,
`A_username` VARCHAR(255) NOT NULL ,
`A_mail` VARCHAR(128)NOT NULL ,
`A_picture` VARCHAR(255) NOT NULL ,
`A_password` VARCHAR(255) NOTNULL ,
PRIMARY KEY (`A_id`)) ENGINE = InnoDB;
```

f. User

Relational Model

User(<u>id</u>, fullname, username, mail, password, city, registration_date, picture)

Functional Dependencies

id-> fullname,username,mail,password,city,registration_date,picture

username->id,fullname,mail,password,city,registration_date,picture

mail->id,fullname,username,password,city,registration_date,picture

password->id,fullname,username,mail,city,registration_date,picture

Candidate Keys

{(id), (username), (mail),(password)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `User` (
`id` INT(32) NOT NULLAUTO_INCREMENT ,
`fullname` VARCHAR(64) NOT NULL ,
`username`VARCHAR(32) NOT NULL ,
`mail` VARCHAR(255) NOT NULL ,
`password`VARCHAR(255) NOT NULL ,
`city` VARCHAR(32) NOT NULL ,
`registration_date` VARCHAR(32) NOT NULL ,
`picture` VARCHAR(255) NOTNULL ,
PRIMARY KEY (`id`),
UNIQUE (`username`),
UNIQUE (`mail`)) ENGINE= InnoDB;
```

g. Moderator

Relational Model

Moderator(u_id, start_date, last_login_date)

Foreign Key u_id references User(id)

Functional Dependencies

u_id-> start_date,last_login_date

Candidate Keys

{(u_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `Moderator` (
`u_id` INT NOT NULL ,
`start_date` VARCHAR(32) NOT NULL ,
`last_login_date` VARCHAR(32) NOTNULL ,
PRIMARY KEY (`u_id`),
FOREIGN KEY (`u_id`) REFERENCES `User`(`id`)ON DELETE RESTRICT
ON UPDATE
RESTRICT) ENGINE = InnoDB;
```

h. RegularUser

Relational Model

RegularUser(u_id, status)

Foreign Key u_id references User(id)

Functional Dependencies

u_id->status

Candidate Keys

{(u_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `RegularUser` (
`u_id` INT NOT NULLAUTO_INCREMENT ,
`status` ENUM('away','online','offline','busy') NOT NULL ,
PRIMARY KEY (`u_id`),
FOREIGN KEY (`u_id`) REFERENCES`User`(`id`) ON DELETE RESTRICT
ON UPDATE RESTRICT) ENGINE = InnoDB;
```

i. Include

Relational Model

Include(T_id, C_id)

Foreign Key T_id references Topic(t_id)

Foreign Key C_id references Category(c_id)

Functional Dependencies

None

Candidate Keys

{(T_id, C_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `Include` ( `T_id` INT NOT NULL , `C_id` INT NOT NULL ,
PRIMARY KEY (`T_id`, `C_id`),
FOREIGN KEY (`C_id`) REFERENCES `Category`(`c_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (`T_id`) REFERENCES `Topic`(`t_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT ) ENGINE = InnoDB;
```

j. Contains

Relational Model

Contains(S_id,T_id)

Foreign Key S_id references Subtopic(S_id)

Foreign Key T_id references Topic(t_id)

Functional Dependencies

None

Candidate Keys

{(S_id, T_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `Contains` ( `S_id` INT NOT NULL , `T_id`INT NOT NULL
, PRIMARY KEY (`S_id`, `T_is`),
REFERENCES `SubTopic`(`S_id`)ON DELETE NO ACTION ON UPDATE
RESTRICT ,
REFERENCES `Topic`(`t_id`) ONDELETE CASCADE ON UPDATE RESTRICT
) ENGINE = InnoDB;
```

k. Modify

Relational Model

Modify(A_id, C_id, date)

Foreign Key A_id references Admin(A_id)

Foreign Key C_id references Category(c_id)

Functional Dependencies

A_id,C_id -> date

Candidate Keys

{(A_id, C_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE  `Modify` ( `A_id` INT NOT NULL ,
`C_id` INT NOT NULL ,
`date` VARCHAR(32) NOT NULL ,
PRIMARY KEY (`A_id`,`C_id`),
FOREIGN KEY (`A_id`) REFERENCES `Admin`(`A_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGNKEY (`C_id`) REFERENCES `Category`(`c_id`) ON DELETE
RESTRICT ON UPDATERESTRICT ) ENGINE = InnoDB;
```

I. CreateTopic

Relational Model

CreateTopic(T_id, U_id)

Foreign Key T_id references Topic(t_id)

Foreign Key U_id references User(id)

Functional Dependencies

None

Candidate Keys

{(T_id, U_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `CreateTopic` ( `T_id` INT NOT NULL ,`U_id` INT NOT
NULL , PRIMARY KEY(`T_id`, `U_id`), REFERENCES `Topic`(`t_id`)
`CreateTopic` ADD FOREIGNKEY (`U_id`) REFERENCES `User`(`id`), )
ENGINE = InnoDB;
```

m. CreateSubtopic

Relational Model

CreateSubtopic(<u>t_id, s_id, u_id</u>)

Foreign Key t_id references Topic(T_id)

Foreign Key s_id references Subtopic(S_id)

Foreign Key u_id references User(id)

Functional Dependencies

None

Candidate Keys

{(t_id, s_id, u_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `CreateSubtopic` ( `t_id` INT NOTNULL ,
`s_id` INT NOT NULL ,
`u_id` INT NOT NULL ,
PRIMARY KEY (`t_id`,`s_id`, `u_id`),
FOREIGN KEY (`t_id`) REFERENCES`SubTopic`(`T_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (`s_id`) REFERENCES `SubTopic`(`S_id`)ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (`u_id`) REFERENCES `User`(`id`) ON DELETE RESTRICT
ONUPDATE RESTRICT) ENGINE = InnoDB;
```

n. HasComment

Relational Model

HasComment(S_id, T_id, Co_id)

Foreign Key S_id references Subtopic(S_id)

Foreign Key T_id references Topic (t_id)

Foreign Key Co_id references Comment(Co_id)

Functional Dependencies

None

Candidate Keys

{(S_id, T_id, Co_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `hasComment` ( `S_id` INT NOT NULL,
`T_id` INT NOT NULL ,
`Co_id` INT NOT NULL ,
PRIMARY KEY (`S_id`,`T_id`, `Co_id`),
FOREIGN KEY (`Co_id`) REFERENCES`Comment`(`Co_id`),
FOREIGN KEY (`S_id`) REFERENCES `SubTopic`(`S_id`),
FOREIGN KEY (`T_id`)REFERENCES `Topic`(`t_id`)
) ENGINE = InnoDB;
```

o. Comtocom

Relational Model

ComtoCom(add_Com, belong_Com)

Foreign Key add_Com references Comment(Co_id)

Foreign Key belong_Com references Comment(Co_id)

Functional Dependencies

None

Candidate Keys

{(add_Com,belong_Com)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `ComtoCom` ( `add_Com` INT NOT NULL ,`belong_Com`
INT NOT NULL , PRIMARY KEY (`add_Com`, `belong_Com`), FOREIGN
KEY (`add_Com`) REFERENCES`Comment`(`Co_id`) FOREIGN KEY
(`belong_Com`) REFERENCES `Comment`(`Co_id`) ) ENGINE =
InnoDB;
```

p. MakeComment

Relational Model

makeComment(Co_id, U_id, date)

Foreign Key Co_id references Comment(Co_id)

Foreign Key U_id references User(id)

Functional Dependencies

None

Candidate Keys

{(Co_id,U_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `makeComment` ( `Co_id` INT NOT NULL ,
`U_id` INT NOT NULL ,
`date` VARCHAR(32) NOT NULL ,
PRIMARY KEY(`Co_id`, `U_id`),
FOREIGN KEY (`Co_id`) REFERENCES`Comment`(`Co_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (`U_id`) REFERENCES `User`(`id`) ON DELETERESTRICT
ON UPDATE RESTRICT) ENGINE = InnoDB;
```

q. LikeComment

Relational Model

likeComment(<u>u_id, Co_id</u>, voteType)

Foreign Key u_id references User(id)

Foreign Key Co_id references Comment(Co_id)

Functional Dependencies

u_id,Co_id -> voteType

Candidate Keys

{(u_id,Co_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `likeComment` ( `u_id` INT NOT NULL ,
`Co_id` INT NOT NULL ,
`voteType` ENUM('like','dislike') NOT NULL ,
PRIMARY KEY (`u_id`, `Co_id`),
FOREIGN KEY (`Co_id`) REFERENCES`Comment`(`Co_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (`u_id`) REFERENCES `User`(`id`) ON DELETE RESTRICT
ON UPDATE RESTRICT ) ENGINE = InnoDB;
```

r. Report

Relational Model

report(<u>Co_id, U_id</u>, date)

Foreign Key Co_id references Comment(Co_id)

Foreign Key U_id references User(id)

Functional Dependencies

Co_id,U_id -> date

Candidate Keys

{(Co_id,U_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE`report` ( `Co_id` INT NOT NULL ,
`U_id` INT NOT NULL ,
`date` VARCHAR(32) NOT NULL ,
PRIMARY KEY(`Co_id`, `U_id`),
FOREIGN KEY (`Co_id`) REFERENCES`Comment`(`Co_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (`U_id`) REFERENCES `User`(`id`) ON DELETERESTRICT
ON UPDATE RESTRICT) ENGINE = InnoDB;
```

s. GiveApproval

Relational Model

giveApproval(U_id, Co_id, date, status)

Foreign Key Co_id references Comment(Co_id)

Foreign Key U_id references User(id)

Functional Dependencies

U_id,Co_id -> date,status

Candidate Keys

{(U_id,Co_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `giveApproval` ( `U_id` INT NOT NULL ,`Co_id` INT NOT
NULL , `date` VARCHAR(255) NOT NULL , `status` VARCHAR(8)
NOTNULL , PRIMARY KEY (`U_id`, `Co_id`), FOREIGN KEY (`Co_id`)
REFERENCES`Comment`(`Co_id`)  FOREIGN KEY (`U_id`) REFERENCES
`User`(`id`) ) ENGINE = InnoDB;
```

t. ApproveTopic

Relational Model

ApproveTopic(U_id, T_id, date, status)

Foreign Key U_id references User(id)

Foreign Key T_id references Topic (t_id)

Functional Dependencies

U_id,T_id ->date,status

Candidate Keys

{(U_id,T_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `ApproveTopic` ( `U_id` INT NOT NULL ,`T_id` INT NOT
NULL , `date` VARCHAR(32) NOT NULL , `status` VARCHAR(255)
NOTNULL , PRIMARY KEY (`U_id`, `T_id`), ADD FOREIGN KEY (`U_id`)
REFERENCES `User`(`id`) ADD FOREIGN KEY(`T_id`) REFERENCES
`Topic`(`t_id`) ) ENGINE = InnoDB;
```

u. BanUser

Relational Model

banUser(U_id, A_id)

Foreign Key U_id references User(id)

Foreign Key A_id references Admin(A_id)

Functional Dependencies

None

Candidate Keys

{(U_id,A_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `banUser` ( `U_id` INT NOT NULL ,
`A_id` INT NOT NULL ,
PRIMARY KEY (`U_id`, `A_id`),
FOREIGN KEY (`A_id`) REFERENCES`Admin`(`A_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT,
FOREIGN KEY (`U_id`) REFERENCES `User`(`id`) ON DELETE RESTRICT
ON UPDATE RESTRICT) ENGINE = InnoDB;
```

v. SuspendUser

Relational Model

suspendUser(A_id, U_id, reactivationDate)

Foreign Key U_id references User(id)

Foreign Key A_id references Admin(A_id)

Functional Dependencies

A_id,U_id -> reactivationDate

Candidate Keys

{(A_id,U_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `suspendUser` ( `A_id` INT NOT NULL,
`U_id` INT NOT NULL ,
`reactivationDate` VARCHAR(32) NOT NULL ,
PRIMARY KEY (`A_id`, `U_id`),
REFERENCES`Admin`(`A_id`) ON DELETE RESTRICT ON UPDATE
RESTRICT,
REFERENCES `User`(`id`) ON DELETERESTRICT ON UPDATE RESTRICT)
ENGINE = InnoDB;
```

w. Follow

Relational Model

follow(following_id, followed_id, date)

Foreign Key following_id references User(id)

Foreign Key followed_id references User(id)

Functional Dependencies

following_id,followed_id->date

Candidate Keys

{(following_id,followed_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `follow` ( `following_id` INT NOT NULL ,
`followed_id` INT NOT NULL ,
`date` VARCHAR(32) NOT NULL ,
PRIMARY KEY(`following_id`, `followed_id`),
FOREIGN KEY (`following_id`, `followed_id`)REFERENCES `User`(`id`,
`id`) ) ENGINE = InnoDB;
```

x. MakesModerator

Relational Model

makesModerator(A_id, U_id)

Foreign Key A_id references Admin(A_id)

Foreign Key U_id references User(id)

Functional Dependencies

none

Candidate Keys

{(A_id,U_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `makesModerator` ( `A_id` INT NOTNULL ,
`U_id` INT NOT NULL ,
PRIMARY KEY (`A_id`, `U_id`) ,
FOREIGN KEY (`A_id`) REFERENCES`Admin`(`A_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT ,
FOREIGN  KEY (`U_id`) REFERENCES `User`(`id`) ON DELETE RESTRICT
ON UPDATE RESTRICT) ENGINE =InnoDB;
```

y. RemoveModerator

Relational Model

removeModerator(A_id, U_id)

Foreign Key A_id references Admin(A_id)

Foreign Key U_id references User(id)

Functional Dependencies

none

Candidate Keys

{(A_id,U_id)}

Normal Form

BSNF

Table Definition

```
CREATE TABLE `removeModerator` ( `A_id` INT NOTNULL ,
`U_id` INT NOT NULL ,
PRIMARY KEY (`A_id`, `U_id`) ,
FOREIGN KEY (`A_id`) REFERENCES`Admin`(`A_id`) ON DELETE
RESTRICT ON UPDATE RESTRICT ,
FOREIGN KEY (`U_id`) REFERENCES `User`(`id`) ON DELETE RESTRICT
ON UPDATE RESTRICT) ENGINE =InnoDB;
```

3. Functional Dependencies and Normalization of Tables

As seen in relational schemes, all relations of this database system are in Boyce-Codd Normal Form (BCNF). Therefore, no decomposition or normalization are necessary for the system. Functional dependencies can be found in the relational schemes part.

4. Functional Components

a. Use Cases / Scenarios

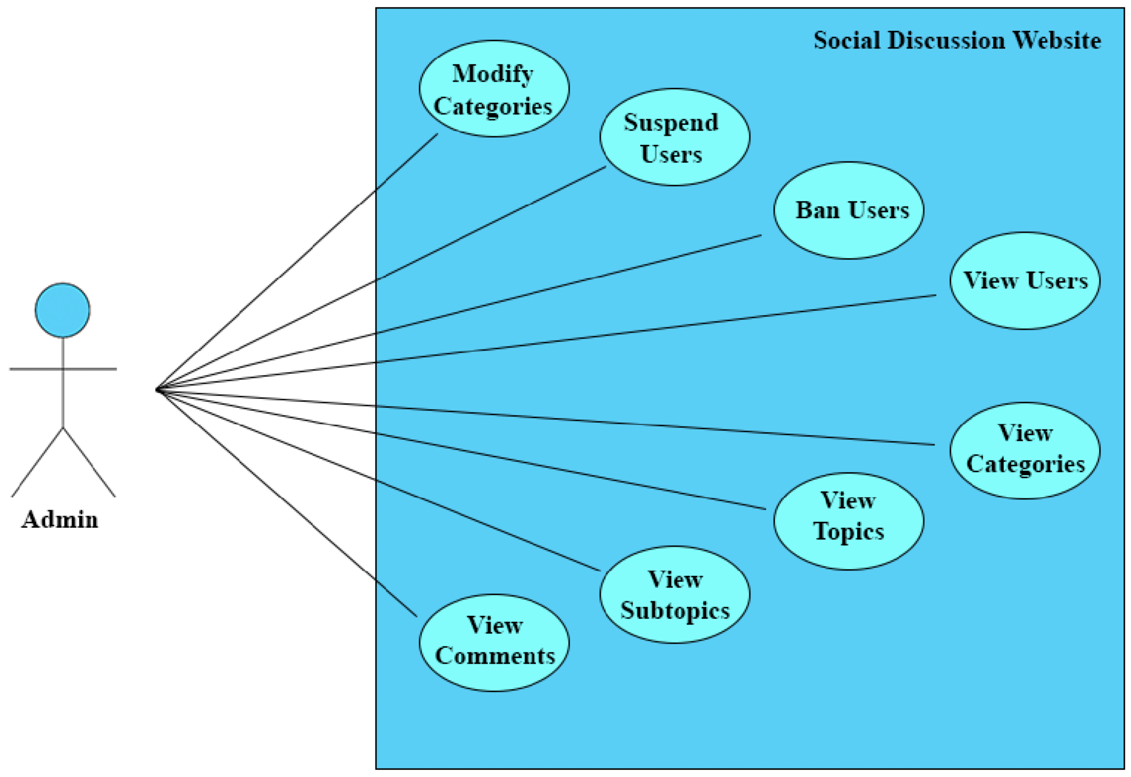
There are four types of users that use this systems: Admin, RegularUser, Moderator and new users. Each type of user have special functions while some share a few functionalities. These functionalities can be seen with specific use case scenarios that are written for each specific type of user. People who do not sign up to the system, will not be able to access contents in the system therefore no other user are available for the system.

i. Admin

The admins are the site administrators, meaning they are the employees of the system who help keep the site up and running. The have special access to functionalities such as banning and suspending users and modifying categories. Other control mechanisms of the system are controlled by moderators.

- **Modify Categories:** Admins are responsible for regulating categories in the system therefore they can modify the categories as they need. Modification date of the categories are stored in the system.
- **Suspend Users:** Admins can suspend users who do not obey rules of this web form for a definite amount of time. When a user is suspended. Their account is deactivated and reactivation of this profile is saved in the system. The system will reactivate this account when the reactivation date comes.
- **Ban Users:** Admins can ban users, who do not obey strict rules of this web form, permanently from the system.
- **View Categories:** Admins can view and search for categories of the system using the search bar or selecting a category.

- **View Topics:** Admins can view and search for topics of the system using the search bar or selecting a category.
- **View Subtopics:** Admins can view and search for subtopics of topics using the search bar or selecting a topic.
- **View Comments:** Admins can view and search for comments of subtopics using the search bar or selecting a subtopic.
- **View Users:** Admins can view and search for users of the system using the search bar or selecting a user.

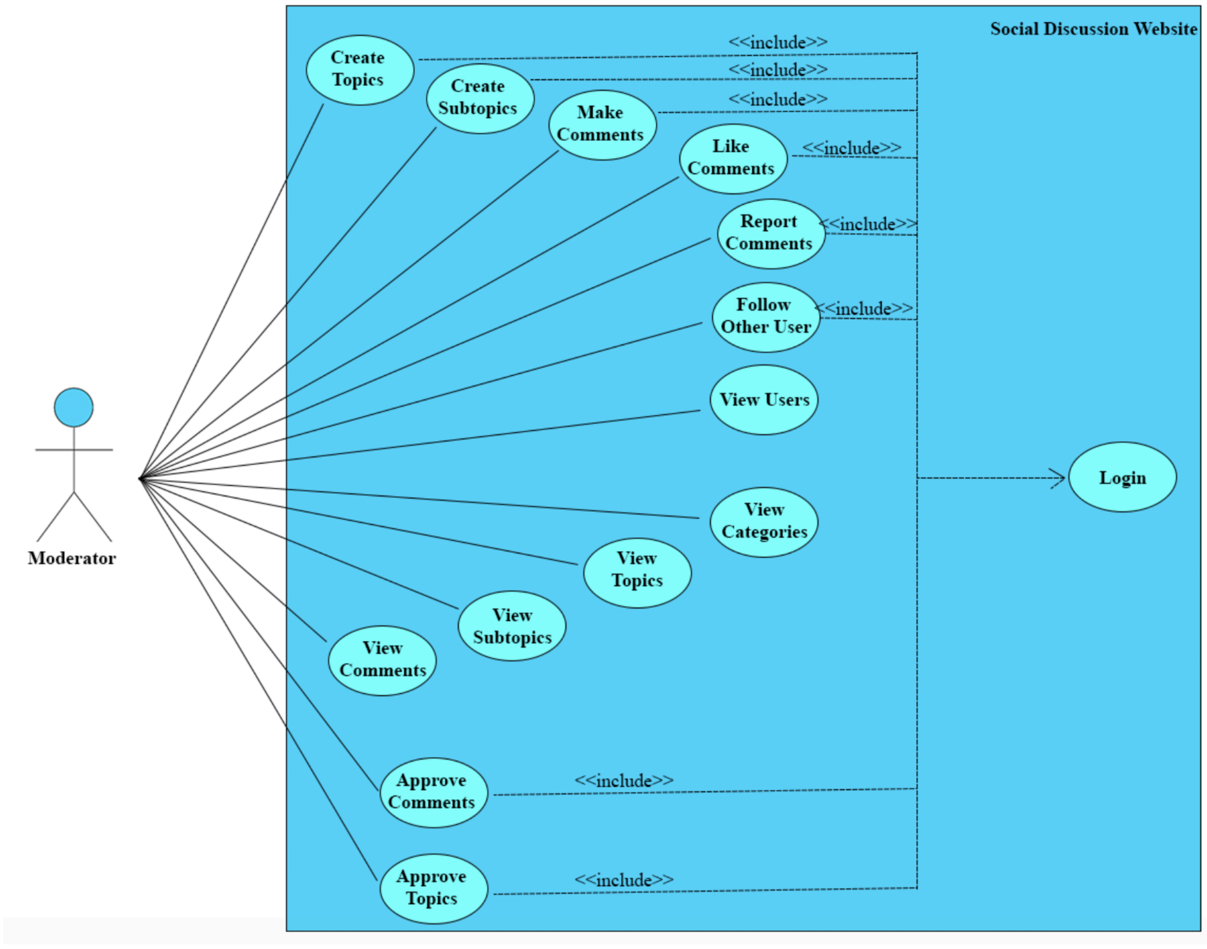


ii. Moderator

The moderators are responsible of approving new comments and subtopics to make them visible on the site. They have special access for these functionalities besides regular user functionalities.

- **Approve Comments:** They are moderating the system by confirming the comments according to comments' appropriateness. Approval or disapproval of the comments and decision date of the moderator are stored in the system. According to moderators decision , the new comment becomes visible on the site or deleted.
- **Approve Topics:** They are moderating the system by confirming the topics according to topics' appropriateness. Approval or disapproval of the topics and decision date of the moderator are stored in the system. According to moderators' decision, the new topic becomes visible on the site or deleted.
- **View Categories:** Moderators can view and search for categories of the system using the search bar or selecting a category.
- **View Topics:** Moderators can view and search for topics of the system using the search bar or selecting a category.
- **View Subtopics:** Moderators can view and search for subtopics of topics using the search bar or selecting a topic.
- **View Comments:** Moderators can view and search for comments of subtopics using the search bar or selecting a subtopic.
- **View Users:** Moderators can view and search for users of the system using the search bar or selecting a user.
- **Create Topics:** Moderators can create topics using the 'Create Topic' button on the main page.

- **Create Subtopics:** Moderators can create subtopics when they are on a specific topic page using 'Create Subtopic' button.
- **Make Comments:** Moderators can make comments when they are on a specific subtopic page using 'Make Comment' option on the page.
- **Like Comments:** Moderators can like comments when they are on a specific subtopic page using 'Like Comment' option on the page.
- **Report Comments:** Moderators can report comments when they are on a specific subtopic page using 'Report Comment' option on the page.
- **Follow Other Users:** Moderators can follow other users when they are on the profile page of the specific user using 'Follow' option on the page.

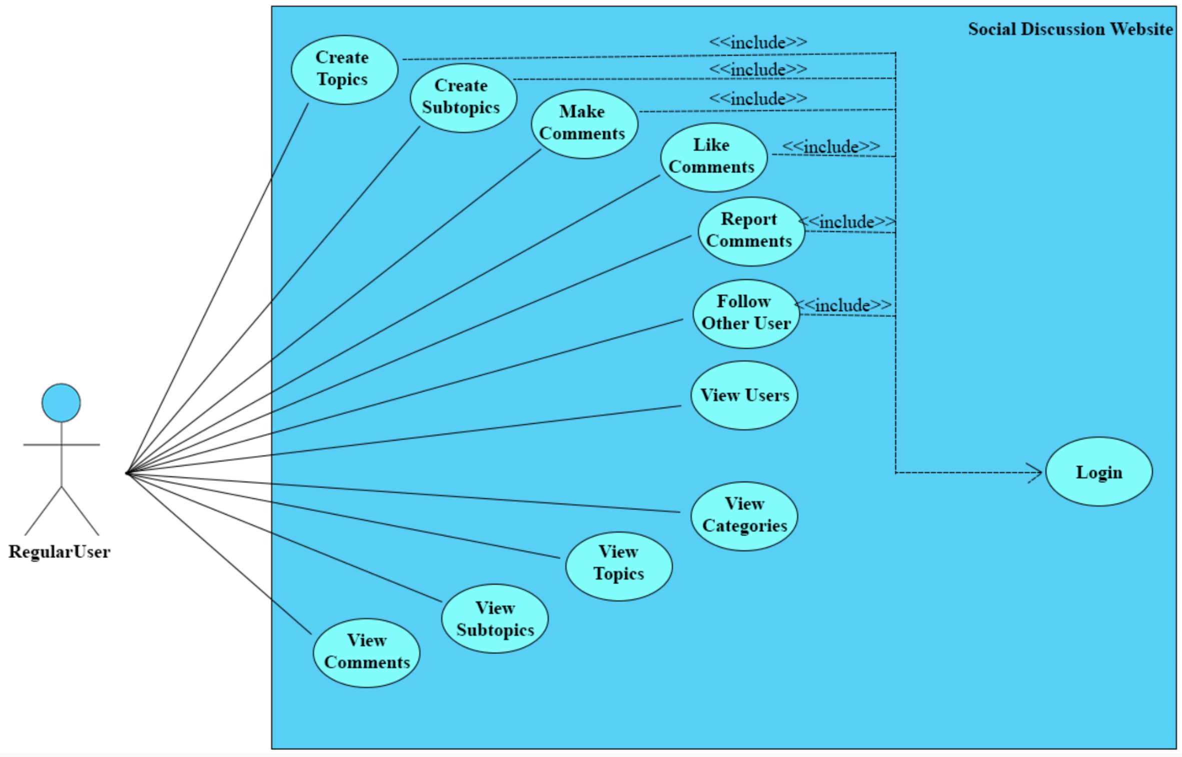


iii. RegularUser

Regular users are the user who enjoy and use this social discussion website. They do not have a job in operations of it therefore they do not have any special use cases that are specific to them. They can view categories, topics, subtopics, comments, users and create topics, subtopics. They can also make, like and report comments and follow other users.

- **View Categories:** Regular Users can view and search for categories of the system using the search bar or selecting a category.
- **View Topics:** Regular Users can view and search for topics of the system using the search bar or selecting a category.
- **View Subtopics:** Regular Users can view and search for subtopics of topics using the search bar or selecting a topic.
- **View Comments:** Regular Users can view and search for comments of subtopics using the search bar or selecting a subtopic.
- **View Users:** Regular Users can view and search for users of the system using the search bar or selecting a user.
- **Create Topics:** Regular Users can create topics using the 'Create Topic' button on the main page.
- **Create Subtopics:** Regular Users can create subtopics when they are on a specific topic page using 'Create Subtopic' button.
- **Make Comments:** Regular Users can make comments when they are on a specific subtopic page using 'Make Comment' option on the page.
- **Like Comments:** Regular Users can like comments when they are on a specific subtopic page using 'Like Comment' option on the page.

- **Report Comments:** Regular Users can report comments when they are on a specific subtopic page using 'Report Comment' option on the page.
- **Follow Other Users:** Regular Users can follow other users when they are on the profile page of the specific user using 'Follow' option on the page.

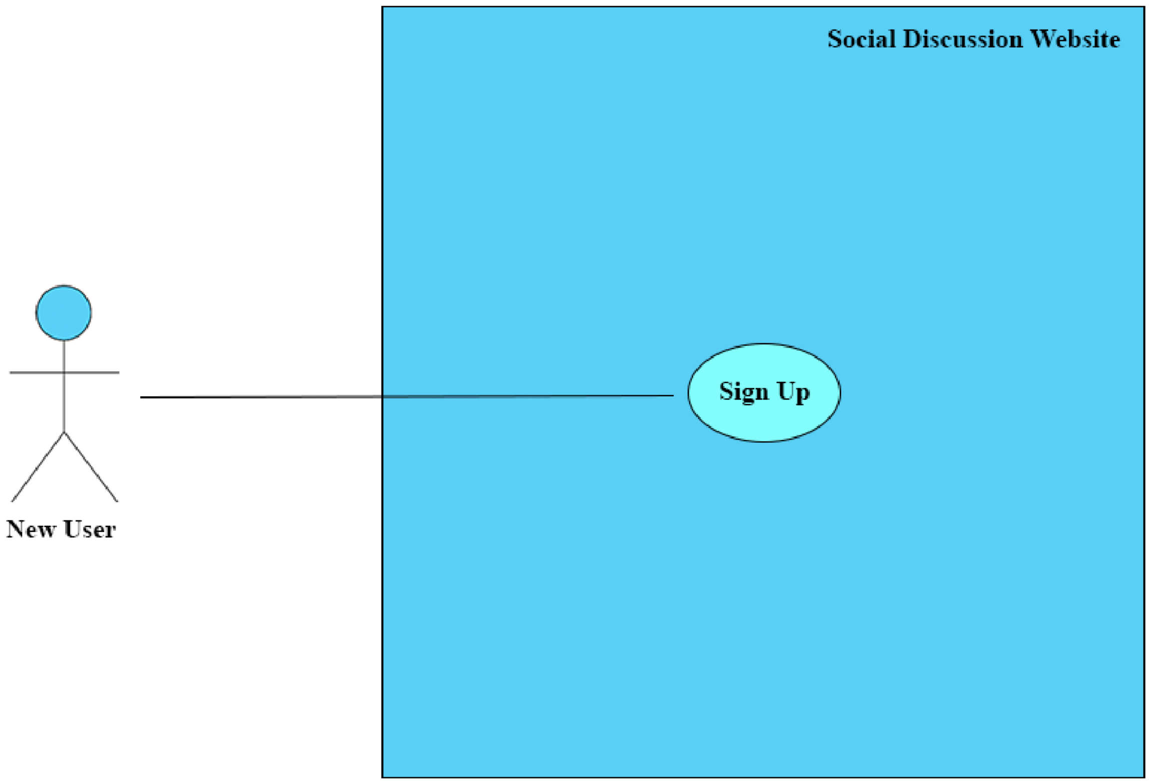


iv. New User

New users are the people who want to sign up to the system. They can use the sign up system to sign up to the system. After signing up new users turn into Regular Users therefore they are no longer considered as a new user.

- **Sign Up:** New user can hit the sign up button to start signing up process. System will load the sign up screen. Then user can enter their information such as name, username, e-mail address, password, city and choose a profile picture for their account, on the screen that comes. After that user can hit the sign up to complete the creation of their account. System will assign a unique Id number to the new user and store the registration date of the user. New user will be created and added to the data table as a regular user using the SQL query below:

Since new users are not yet users of the system, they cannot access the contents of the system therefore they do not have any functions other than sign up function.



b. Algorithms

1-Search Related Algorithm

User interacting with the application can search through the website with keywords. The user can do this in several ways . There are 4 options for user can select to search on the system such as by Categories, Topics, Subtopics or/and Users. For instance, user who wants to search in topics and subtopics needs to select these two check buttons. That is, user can use every combination of these four. Only restriction is user who wants to search have to at least one option among them.

2-Logical Requirements

Logical errors should be handled and prevented in order the system to work properly. System should be designed such a way that total topic number cannot be more or less than sum of the Topic that are held in the specific Category, which means there cannot a topic without a Category. Same procedure should be satisfied for Subtopics.

Reactivation date of the suspended user cannot be chosen before or as the current date by moderator in order to prevent nepotism.

In order to check and ease the storage of the dates, we choose to store date as domain type varchar not in date, time or datetime in order to prevent storing in different types of date . We choose to store each date as varchar to manipulate, check. For instance, reactivation_date inside the suspendedUser table and registration_date inside the User table and all of the dates inside all the tables should be on the same format. Due to the fact that we cannot obtain easily required format for domain type date, time and datetime from the user registration form, we find this way storing date is more manipulable and easier.

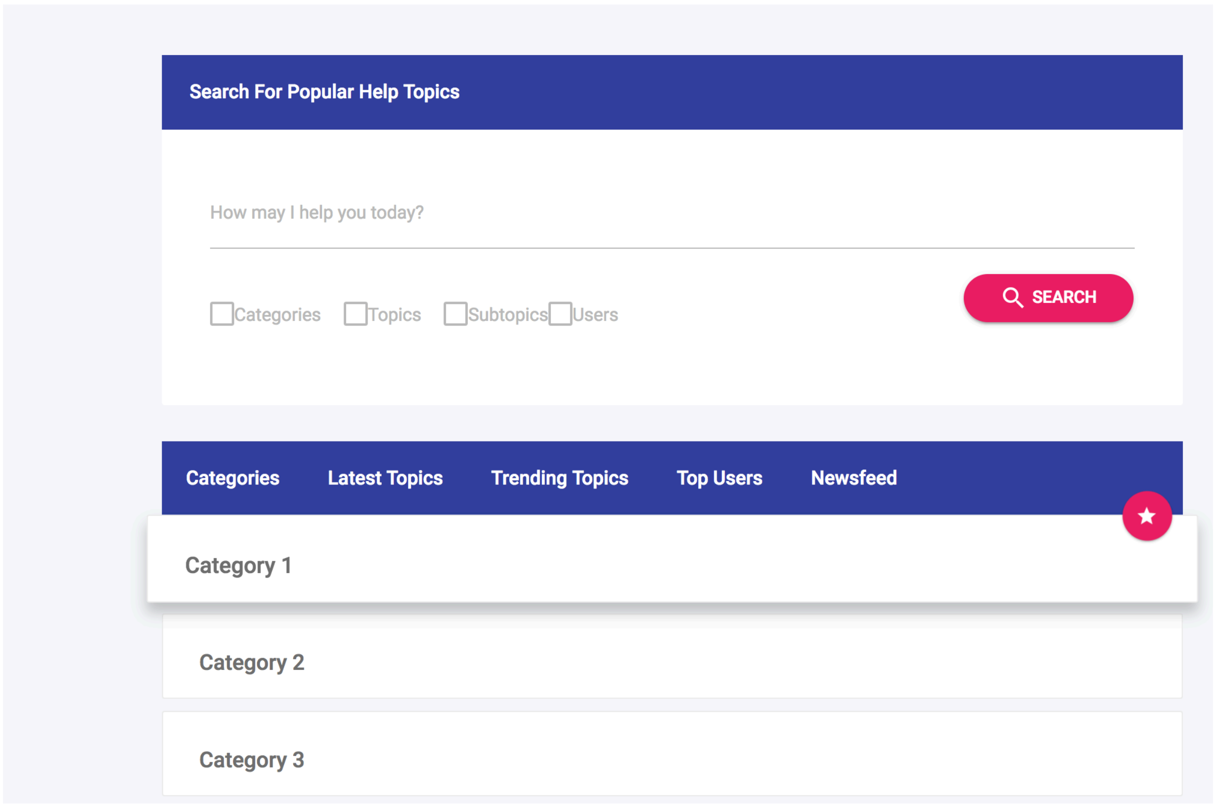
c. Data Structures

In the database, relational schemes are created using two types of variables. These are Numeric types and text types.

- Numeric types are used for keeping numeric data such as IDs, counts etc.. Numeric types are created as INT as standart numeric type.
- Other informations that require text types such as name, city names, paswords etc., are created as VARCHAR as standart.

5. User Interface Design and Corresponding SQL Statements

a. HomeScreen



Getting top matching categories for the query field of the homepage

Create view top10Categories (C_id, C_name, T_Name) as

Select C_id, C_name, T_name

From top10Categories Natural Join Category Natural Join Include Natural Join Topic

Group by C_id

Having count(T_id) <= 3

With top10Categories (C_id) as

Select C_Id

From Category

Where C_name like '%$input'% C_id in (

```
                    Select  S1.C_id

                    From sumTotal S1

                    Where (select(count S2.sum)

                              From sumTotal S2

                              Where S2.sum >  S1.sum ) <= 9 )


With sumTotal (C_id,sum) as

           Select C1.C_id, (C1. Number_of_subtopics + C1. Number_of_Topics) as sum

          From Category C1

          Group by C1.C_id
```

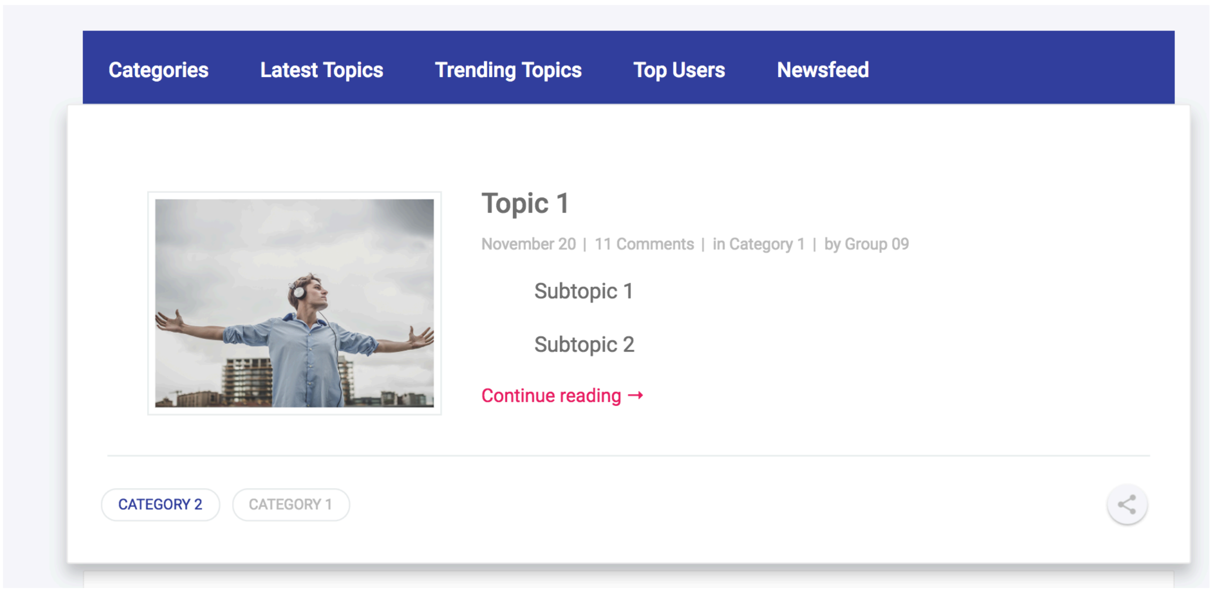
Getting top matching topics for the query field of the homepage

```
Create view top10Topics (T_id,T_name, S_name) as

        Select T_id, T_name, S_name

        From top10Topics Natural Join Contains Natural Join Subtopic

        Group by T_id

        Having count(T_id) <= 3



With top10Topics(_id) as

        Select T_Id

        From Topic

        Where  T_name like '%$input'% T_id in  (

        Select  S1.T_id

        From Topic S1

        Where (select(count S2.Number_of_subtopics)

                From Topic S2

                Where S2.Number_of_subtopics >  S1.Number_of_subtopics) <= 9 )
```

Getting top matching Users for the Query Section

```
With countFollowers(U_id, counter) as
        Select U_id, count(followed_id)
        From User Natual Join Follow
        Group by U_id


Create view top10Users (U_id,U_name, U_username, followerCount, U_registration_date,
U_picture) as
        Select (U_id,U_name, U_username, followerCount, U_registration_date, U_picture)
        From User Natural Join countFollowers
        Where  U_name like '%$input'% U_id in  (
                Select  S1.U_id
                From countFollowers S1
                Where (select(count S2.counter)
                        From countFollowers S2
                        Where S2.counter >  S1.counter) <= 9 )
```

b. Latest Topic Screen



Topic 1

November 20 | 11 Comments | in Category 1 | by Group 09

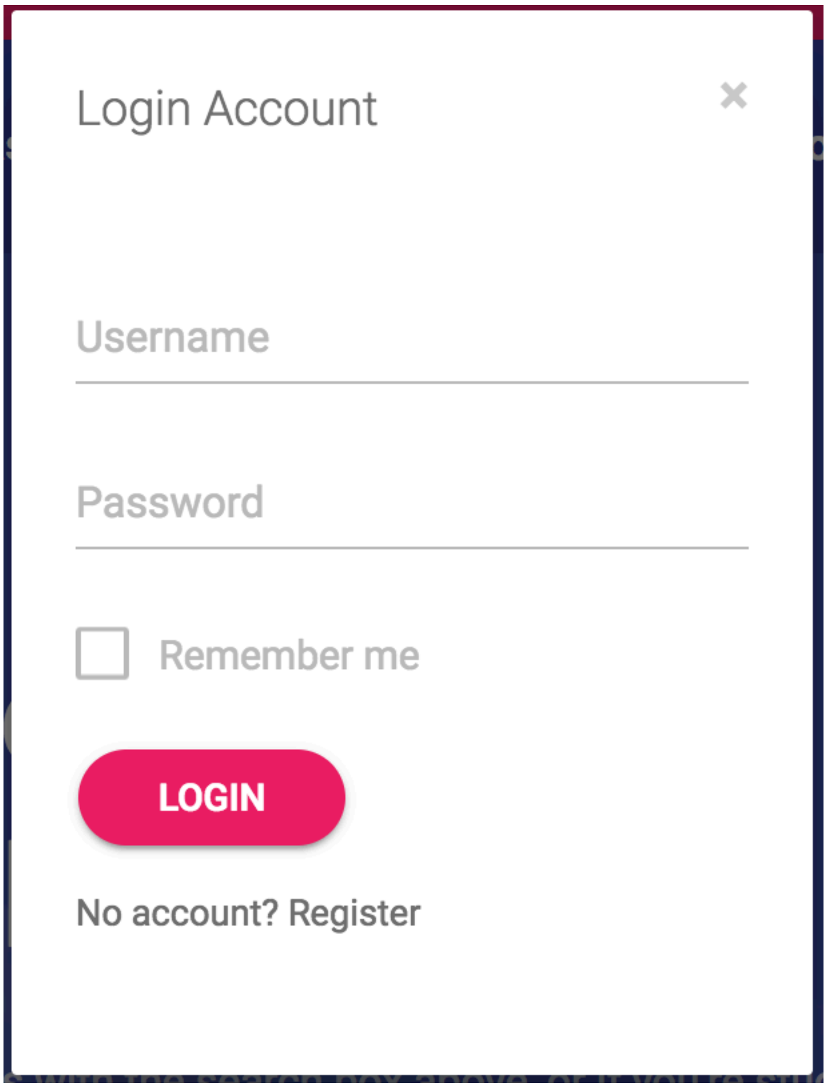
Subtopic 1

Subtopic 2

Getting Latest Topics for the LatestTopics tab in the HomePage

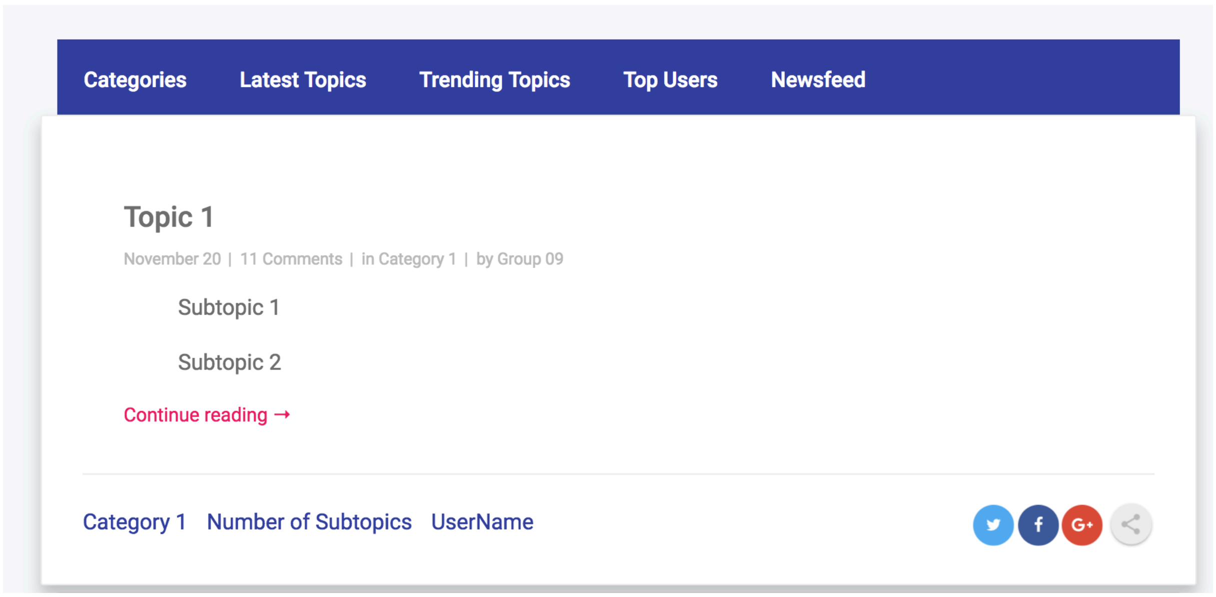
```
Create view latestTopics (T_id,T_name, createDate, Number_of_Subtopics,

T_description,T_icon) as

        Select T_id,T_name, createDate, Number_of_Subtopics, T_description,T_icon

        From Topic

        Where  T_id in  (

                Select  S1.T_id

                From Topic S1

                Where (select(count S2.T_id)

                        From Topic S2

                        Where S2.T_createDate >  S1.createDate) <= 9 )
```

c. Login Screen



```
SELECT *

FROM person

WHERE username = @username AND password =@password
```

d. Newsfeed Screen



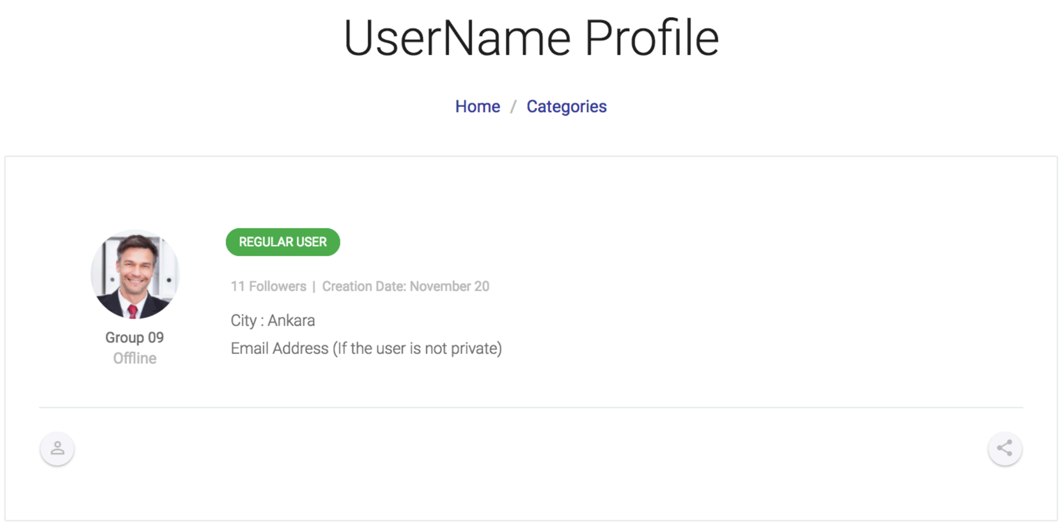
Getting NewsFeed tab in the HomePage:

```
With countFollowing(U_id, counter) as

        Select U_id, count(following_id)

        From User Natual Join Follow

        Group by U_id



Create view Newsfeed (U_id,U_name, U_username, T_id,T_name, createDate,

Number_of_Subtopics, T_description,T_icon) as

        Select U U_id,U_name, U_username, T_id,T_name, createDate,

Number_of_Subtopics, T_description,T_icon

        From Users Natural Join createTopic as A Natural Join Topic

        Where  U_id in  (

                Select  S1.U_id

                From countFollowing S1

        )
```

Group by U_id

LIMIT(3)

e. User Profile Screen


UserName Profile

Home / Categories

REGULAR USER

11 Followers | Creation Date: November 20

City : Ankara

Email Address (If the user is not private)

Group 09

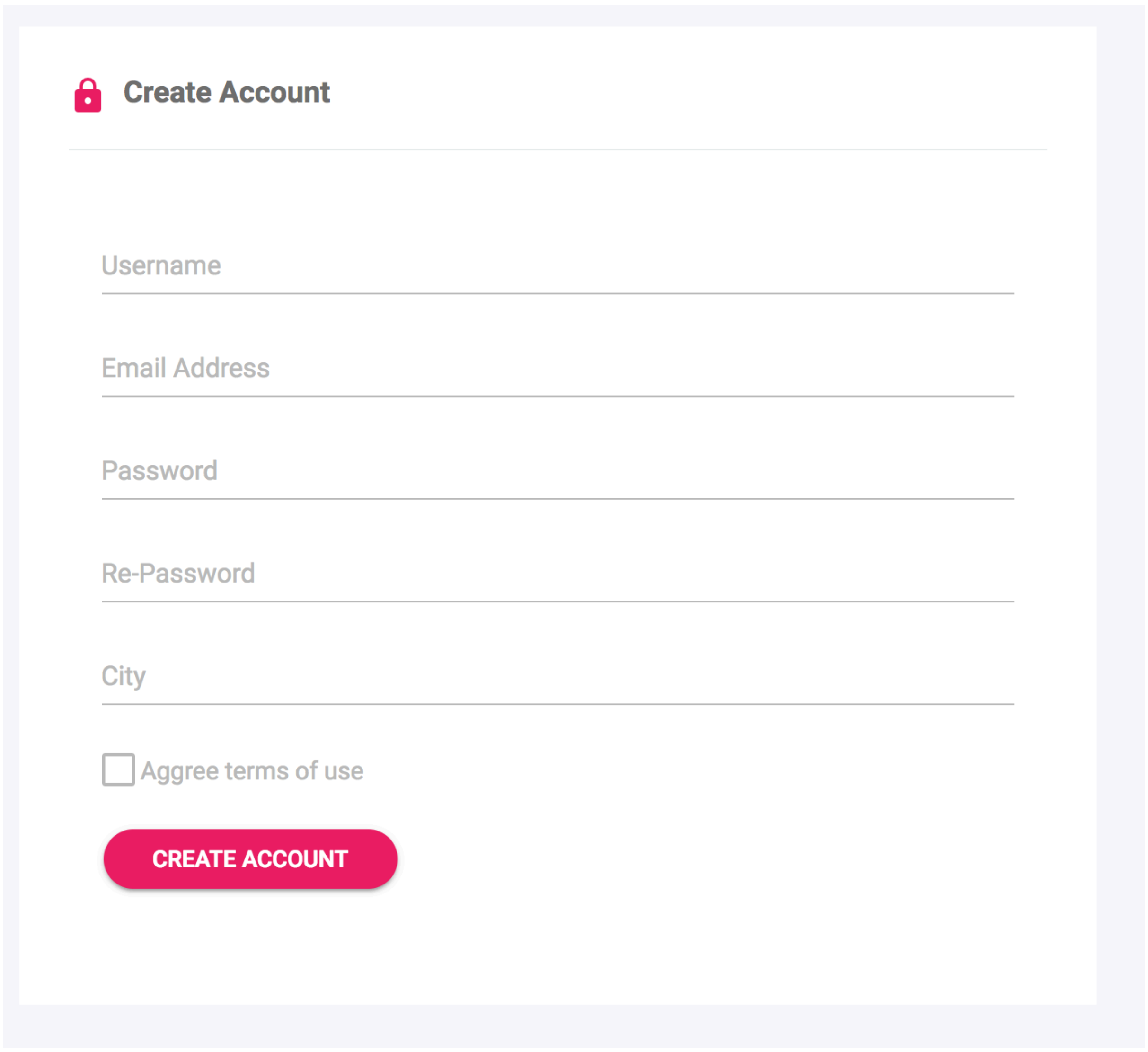
Offline

Select *

From User Natural Join countFollowers

Where U_username = @username

f. Registration Screen


Create Account

Username

Email Address

Password

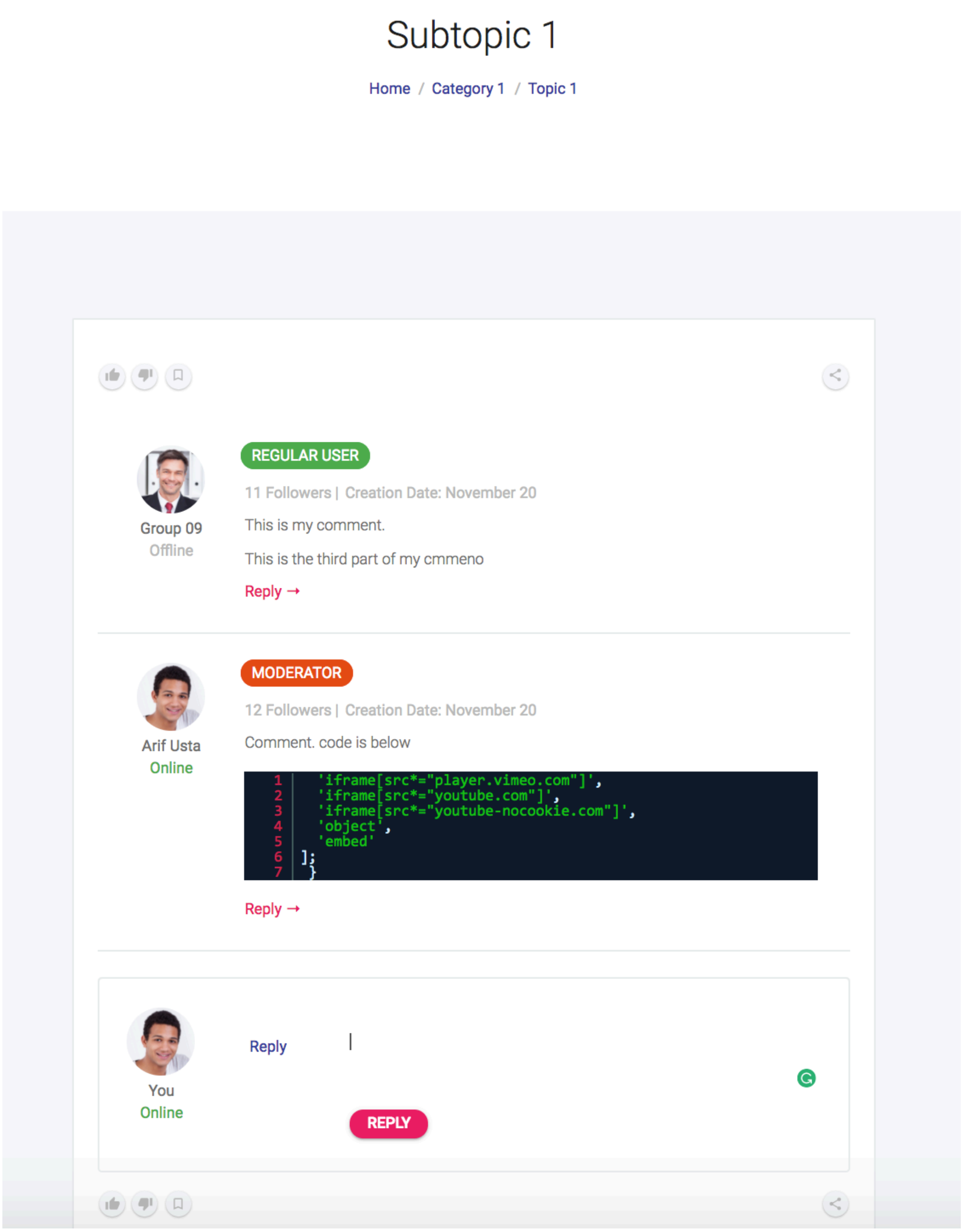
Re-Password

City

☐ Aggree terms of use

```
INSERT INTO RegularUser

VALUES (@username, @email, @password, @repassword, @city)

WHERE @password = @repassword
```

g. Subtopic Page Screen



Subtopic 1

Home / Category 1 / Topic 1

REGULAR USER

Group 09
Offline

11 Followers | Creation Date: November 20

This is my comment.

This is the third part of my cmmeno

Reply →

MODERATOR

Arif Usta
Online

12 Followers | Creation Date: November 20

Comment. code is below

```
    'iframe[src*="player.vimeo.com"]',
    'iframe[src*="youtube.com"]',
    'iframe[src*="youtube-nocookie.com"]',
    'object',
    'embed'
  ];
  }
```

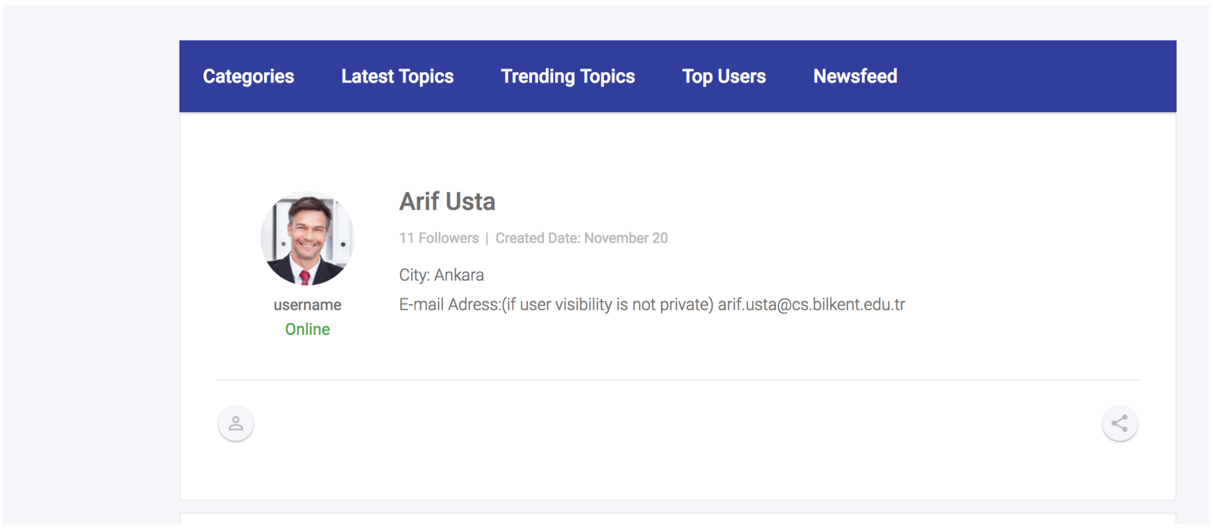
Reply →

You
Online

Reply

REPLY

h. Top Users Screen



Arif Usta

11 Followers | Created Date: November 20

City: Ankara

E-mail Adress:(if user visibility is not private) arif.usta@cs.bilkent.edu.tr

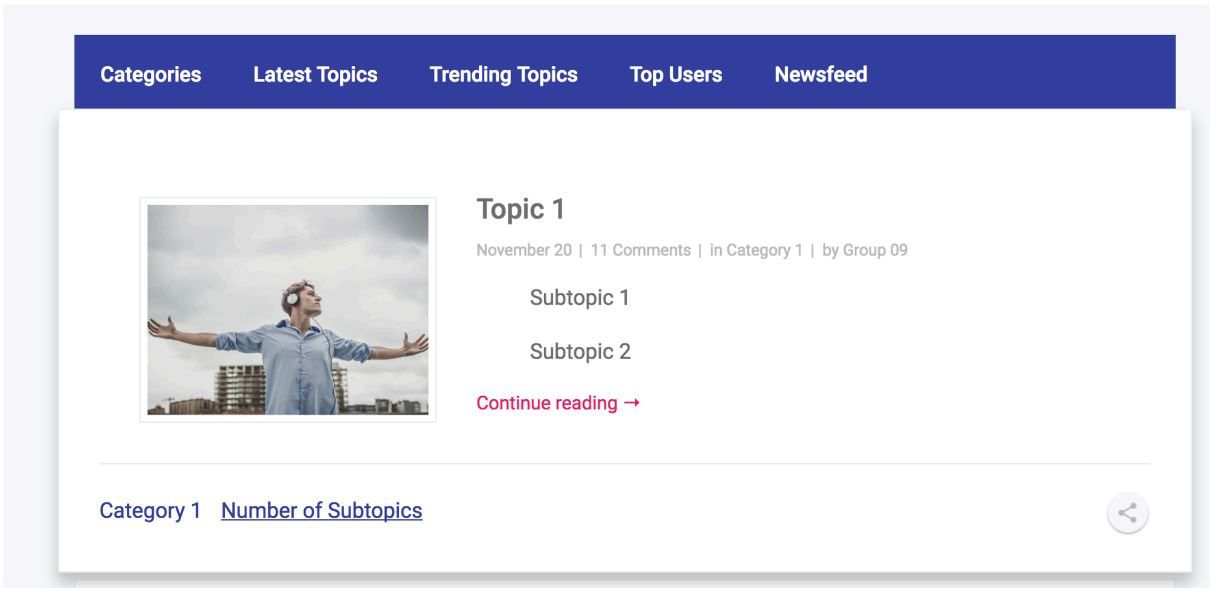
username

Online

Getting Top Users tab in the HomePage

```
Create view topUsers (U_id,U_name, U_username, U_mail, U_registration_date, U_picture)

as

        Select U_id,U_name, U_username, U_mail, U_registration_date, U_picture

        From User Natural Join countFollowers

        Where  U_id in  (

                Select  S1.U_id

                From countFollowers S1

               Where (select(count S2.counter)

                        From countFollowers S2

                        Where S2.counter >  S1.counter) <= 9 )
```

i. Trending Topics Screen



Topic 1

November 20 | 11 Comments | in Category 1 | by Group 09

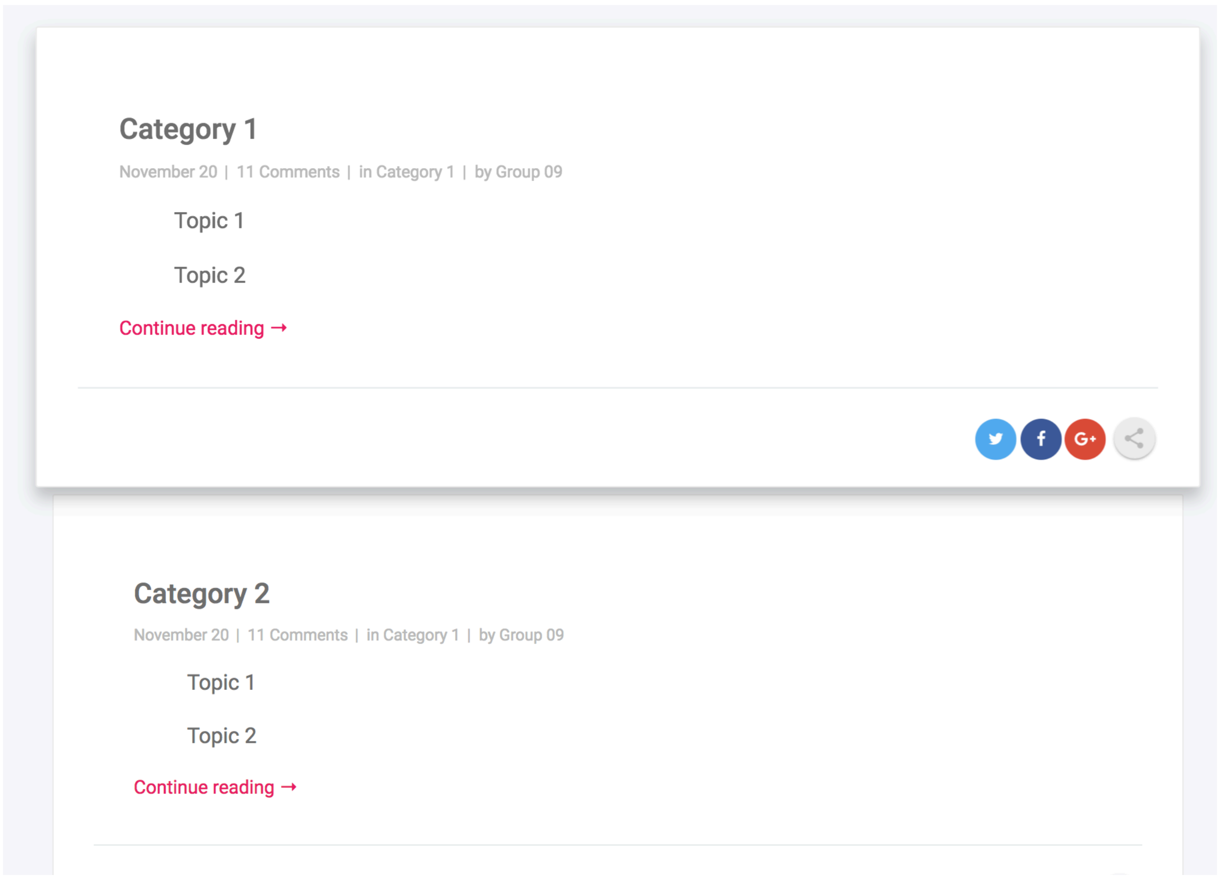
Subtopic 1

Subtopic 2

Continue reading →

Category 1 Number of Subtopics

j. Categories Page Screen


Category 1

November 20 | 11 Comments | in Category 1 | by Group 09

Topic 1

Topic 2

Continue reading →

Category 2

November 20 | 11 Comments | in Category 1 | by Group 09

Topic 1

Topic 2

Continue reading →

This page follows the same algorithm as the Topic selection query.

6. Advanced Database Components

a. Views

This section will make use of the views that were presented on the queries of section 5. Our project will make use of four main views that define the functionalities of our application.

- The topUsers view will make visible the information of the users who are the most followed in our system (a measure of popularity) and they will be ranked on pagination that shows 10 users per page. Based on this information other users may follow these popular users within our application.

- The latestTopics view will be used to show paginations of topics that have been most recently posted in the application by other users in the database who are not necessarily followed by the user whom this information is being presented to. Each topic will show by default the headers of three subtopics for which the user can go and leave the comment.

- The NewsFeed view will provide information for the users that you follow starting from the topics they may have created recently or comments that they may have made. The information displayed here will incorporate the basic information for the user such as the createDate of his account the number of followers he has and the comment of the topic the user replied to or created.

b. Stored Procedures

During the filtering of the pages we will need to store several procedures that will make several count processes easier and more generic. The first such procedure is countFollowing which gives the number of the users followed by each specific user.

```
With countFollowing(U_id, counter) as
Select U_id, count(following_id)
From User Natual Join Follow
```

Similarly the number of the followers for each user will be frequently computed therefore the below we have presented a procedure to get the id of each user and the corresponding count of followers that the user has . This information will be used when fetching the user Profile page and the Subtopic page along with the comments that pertain to each user.

```
With countFollowers(U_id, counter) as
Select U_id, count(followed_id)
From User Natual Join Follow
Group by U_id
Group by U_id
```

c. Reports

1. User Stats Reports

When an admin wants to see the stats of how many subtopics, topics created and how many comments are made he/she can see them as a report. Admin also sees the status of each user.

d. Triggers

- When a Topic is deleted from the system, subtopic(s) and comment(s) that are included in it will be deleted.
- When a user banned, his/her all comment(s) will be deleted but later usage, topic(s) and subtopic(s) of the banned user will be owned by the random moderator on the system.
- When a subtopic is deleted, all comment(s) under that will also be deleted.
- When a subtopic or topic is added to the system, attributes inside the Category table namednumber_of_topics and number_of_subtopics will be increased or when a topic or subtopic is deleted they will be decreased relatively.
- When administrator makes regular user a moderator, regular user will pass from RegularUser table to Moderator table.
- When a user likes/dislikes/reports a comment, relative column for a specific comment will be increased inside the Comment table.
- When a subtopic is added, number_of_subtopic column for a specific topic will be increased by 1 inside the Topic table.
- When admin adds a moderator, relative column for that moderator start_date will be changed with the current date when he/she added.

e. Constraints

- The total counts of subtopics inside the Topic table cannot be less than or more than counts of the subtopics.
- There cannot be any two same named Category.
- When a user is banned, he/she cannot use system anymore.

7. Implementation Plan

The Core of our system will be handled by the use of MySQL database and phpMyAdmin. The website itself however will require the use of some php javascript and HTML. These are put in a small amount and they will only affect the layout of the application not the underlying system by itself.

8. Website

All the stages of this project will be documented in a public repository of the following github user profile:

- https://github.com/aldotali

The full link to the report is the following:

- https://github.com/aldotali/CS353-Database-Systems/blob/master/DesignReport/GROUP%209%20Design%20Report.pdf