Database Systems

CS353

Project Design Report

Group 6

Ahmet Emre Nas   **21402357**

Doğukan Altay   **21400627**

Umut Akös   **21202015**

**Batıkan Hayta**   **21301382**

Contents

1. Brief Description

Our application Tidder aims to connect every person all around word to discuss any topic a person can think about. A user can create any topic about a question a, discussion or a news. Other users of the application can comment to these topics. Even though these topics are public and visible to anyone, our application provide private message system between users. People can write to each other privately by this system. People need to register in order to post a topic or a comment. They will login into website by their username and passwords. Ever user has their own profile page where they can write about themselves to introduce who they are to other users.

This report contains, revised ER diagram, relation schemas, mock-ups for user interface and database components of Tidder Application.

Differences between revised E/R diagram and proposal E/R diagram

- Moderator entity connected to topic entity with relation diamond
- Send Diomand converted to mesEvent which holds for date users'id and message's id
- Message entity contains PK, text and subject
- Activity history deleted since it's basically same with activity
- Activity_done now has two attributes for user id and activity id
- Topic, subtopic, post and comment entites added in a package
- Subcomment diomand added to comment with two comment id attributes
- A new diamond between subtopic and post added which holds ids for both entities
- A new diamond between post and comments added which holds ids for both entities

2. Revised E/R Diagram
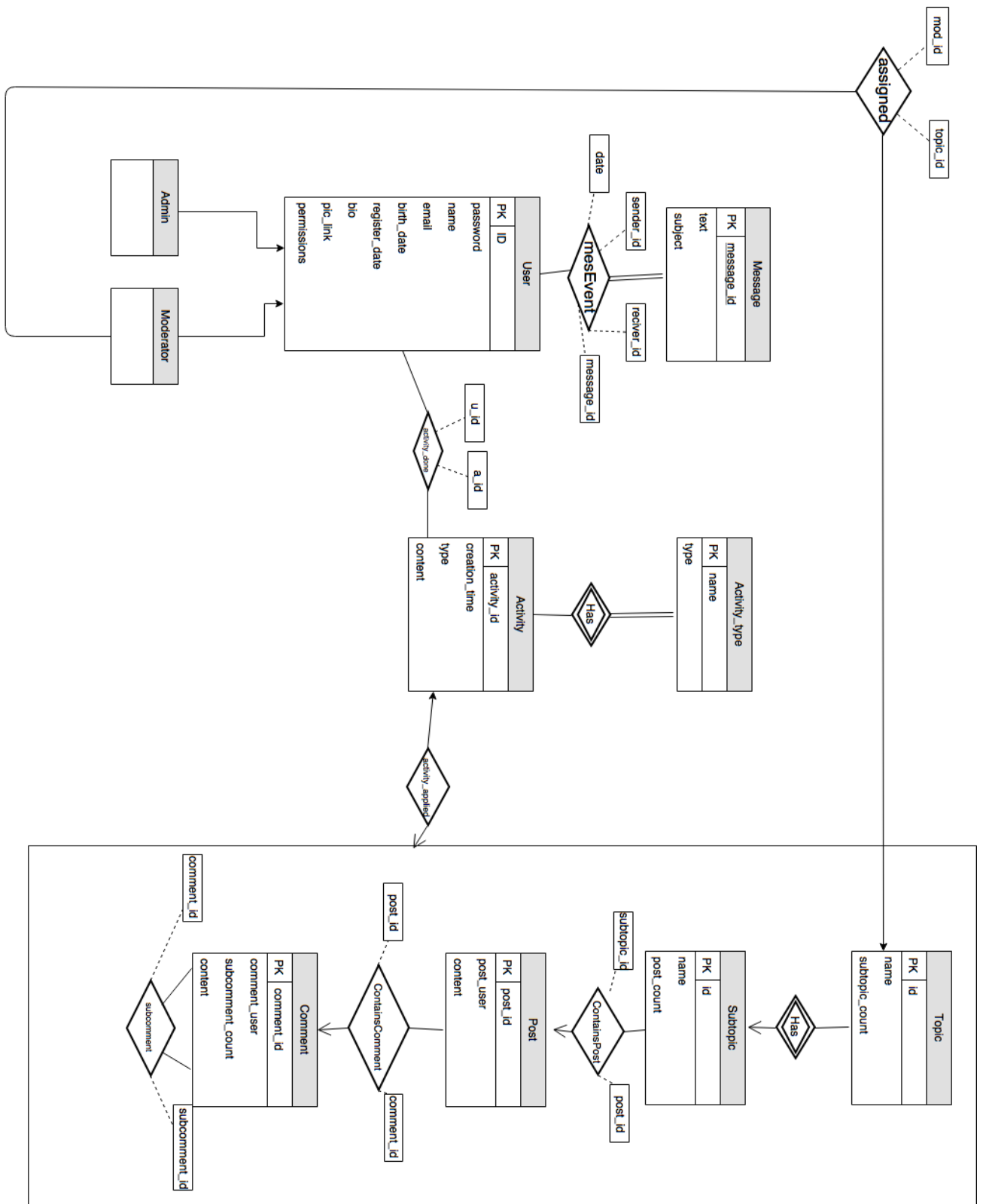
3. Relation Shema

3.1.1 User

Relational Model

user(ID, password, name, email, birth_date, register_date, bio , pic_link)

Primary Key -> ID

Candidate Key -> ID

Functional Dependencies

ID -> password, name, email, birth_date, register_date, bio, pic_link

Normal Form

BCNF

Table Definition:

```
CREATE TABLE User(
        ID int,
        password VARCHAR(30) not null,
        name VARCHAR(20) not null,
        email VARCHAR (50) NOT NULL,
        birth_date DATE,
        register_date DATE,
        bio VARCHAR(350)
        pic_link VARCHAR(100)
);
```

3.1.2 Message

Relational Model

message(ID, text, subject**)**

Primary Key -> ID

Candidate Key -> ID

Functional Dependencies

ID -> text, subject

Normal Form

BCNF

Table Definition:

CREATE TABLE User(

      ID int,

      text VARCHAR(1000) NOT NULL,

      subject VARCHAR (50)

);

3.1.3 mesEvent

Relational Model

mesEvent(<u>message_id, sender_id, receiver_</u>id, date)

foreign key message_id references message(id)

foreign key sender_id references user(id)

foreign key receiver_id references user(id)

Normal Form

BCNF

Table Definition:

```
CREATE TABLE mesEvent(
        message_id int,
        sender_id int,
        receiver_id int,
        FOREIGN KEY(message_id) REFERENCES message(id),
        FOREIGN KEY(sender _id) REFERENCES user(id)
        FOREIGN KEY(receiver _id) REFERENCES user(id)
) Engine = InnoDB;
```


3.1.4 Assigned

Relational Model

assigned(<u>mod_id, topic_id</u>)

foreign key mod _id references mod(id)

foreign key topic _id references topic(id)

Normal Form

BCNF

CREATE TABLE mesEvent(

      mod _id int,

      topic _id int,-

      FOREIGN KEY(mod _id) REFERENCES message(mod),

      FOREIGN KEY(topic _id) REFERENCES user(topic)

) Engine = InnoDB;

3.1.5 Activity

Relational Model

activity(<u>activity _id</u>, creation_time,_type, content)

PRIMARY KEY -> activity_id

CANDIDATE KEY -> activity_id

Functional Dependencies

Activity_id -> creation_time, type, content

Normal Form

BCNF

Table Definition:

CREATE TABLE activity(

      activity _id int NOT NULL,

      creation_time DATE NOT NULL,-

      type int,

      content VARCHAR(1000)

) Engine = InnoDB;

3.1.6 Activity_type

Relational Model

Activity_type(<u>name</u>, type**)**

PRIMARY KEY -> name

CANDIDATE KEY -> name

Functional Dependencies

name -> type

Normal Form

BCNF

Table Definition:

CREATE TABLE activity_type(

      Name VARCHAR(20) NOT NULL,

      type int NOT NULL

) Engine = InnoDB;

3.1.7 Topic

Relational Model

topic(<u>id</u>, name, subtopic_count**)**

PRIMARY KEY -> id

CANDIDATE KEY -> id

Functional Dependencies

id -> name, subtopic_count

Normal Form

BCNF

Table Definition:

```
CREATE TABLE topic(
        id int NOT NULL,
        name  VARCHAR(30) NOT NULL,
        subtopic_count int
) Engine = InnoDB;
```

3.1.8 Subtopic

Relational Model

Sub_topic(id, name, post_count)

PRIMARY KEY -> id

CANDIDATE KEY -> id

Functional Dependencies

id -> name, post _count

Normal Form

BCNF

Table Definition:

```
CREATE TABLE topic(
        id int NOT NULL,
        name  VARCHAR(30) NOT NULL,
        post _count int
) Engine = InnoDB;
```

3.1.9 post

Relational Model

post (post_id, post_user ,content)

PRIMARY KEY -> post_id

CANDIDATE KEY -> post_id

Functional Dependencies

post_id -> post_user, content

Normal Form

BCNF

Table Definition:

CREATE TABLE post(

       post_id int NOT NULL,

       post_user int NOT NULL,

       content VARCHAR(1000) NOT NULL

) Engine = InnoDB;

3.1.10 Comment

Relational Model

comment (<u>comment id,</u> comment_user , subcomment_count, content)

PRIMARY KEY -> comment_id

CANDIDATE KEY -> comment_id

Functional Dependencies

comment id -> comment user, subcomment_count, content

Normal Form

BCNF

Table Definition:

CREATE TABLE comment(

      comment _id int NOT NULL,

      comment _user int NOT NULL,

subcomment_count int,

content VARCHAR(1000) NOT NULL

) Engine = InnoDB;

3.1.11 ContainsPost

Relational Model

containsPost (subtopic_id, post_id)

FOREIGN KEY -> subtopic_id references subtopic(id)

FOREIGN KEY -> post_id references post(id)

Normal Form

BCNF

Table Definition:

CREATE TABLE ContainsPost(

post _id int NOT NULL,

subtopic_id int NOT NULL,

FOREIGN KEY (subtopic_id) REFERENCES subtopic(id)

FOREIGN KEY (post_id) REFERENCES post(id)

) Engine = InnoDB;

3.1.12 ContainsComment

Relational Model

containsComment (comment_id, post_id)

FOREIGN KEY -> comment_id references comment(id)

FOREIGN KEY -> post_id references post(id)

Normal Form

BCNF

Table Definition:

CREATE TABLE ContainsComment(

comment_id int NOT NULL,

post_id int NOT NULL,

FOREIGN KEY (comment_id) REFERENCES comment (id)

FOREIGN KEY (post_id) REFERENCES post(id)

) Engine = InnoDB;

3.1.13 subComment

Relational Model

subComment (comment_id, subcomment_id)

FOREIGN KEY -> comment_id references comment(id)

FOREIGN KEY -> subcomment_id references comment(id)

Normal Form

BCNF

Table Definition:

CREATE TABLE subComment(

comment_id int NOT NULL,

subcomment_id int NOT NULL,

FOREIGN KEY (comment_id) REFERENCES comment (id)

FOREIGN KEY (subcomment_id) REFERENCES comment(id)

) Engine = InnoDB;

4. Functional Companents

4.1 Algorithms

4.1.1 Search Algorithm

Search algorithm will be text based search algortihm. When a user try to search something all databases will be traced in order to find most suitable results. It will also include a parser since user may want to search only in posts or comments. For instance "post: What do you guys think about new iPhone X" will only search only in posts.

4.1.2 Password Encryption

Password's of users will be encrypted in order to increase security in case of an security issues. Passwords will be converted to one way encrypted data using hash values. Therefore not even devolopers can see user's passwords.

4.2 Use Cases

Tidder has three type of user which are normal user, admin and mod.

4.2.1 Normal User
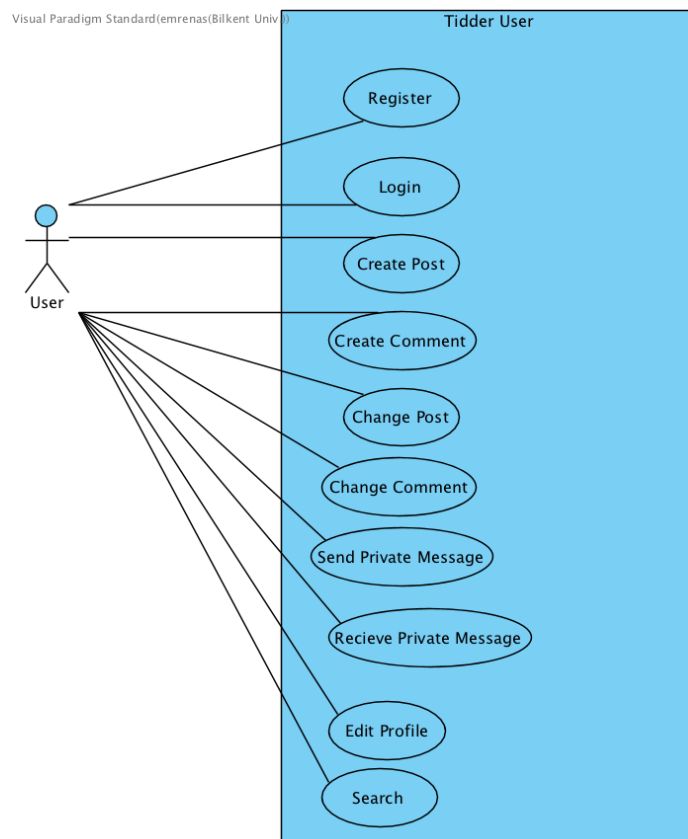
- Register: Users register by providing a username and a password
- Login: Users login by typing their username and password
- Create Post: User can create any post in any topic
- Change Post: User can change their own post
- Change Comment: User can change their own comment
- Create comment: User create a comment to a post or to a comment
- Send Private Message: User send private message to other users

- Receive Private Message: User receive private messages which sended to them by other users.
- Search: Admin can search anything in the application.

One of the actor of Tidder is user. User can access the software by log inning with their password and usernames. Password and username are stored in database where passwords are encrypted by hash function in order to increase security. If they logged in to software properly they can start to create a post or comment. If they want they can send private message to other users. Also, they can edit their profile page in order to introduce themselves to other users. If they do not want to create a post or comment they can search between posts to find particular topics they seek for.

4.2.2 Admin User
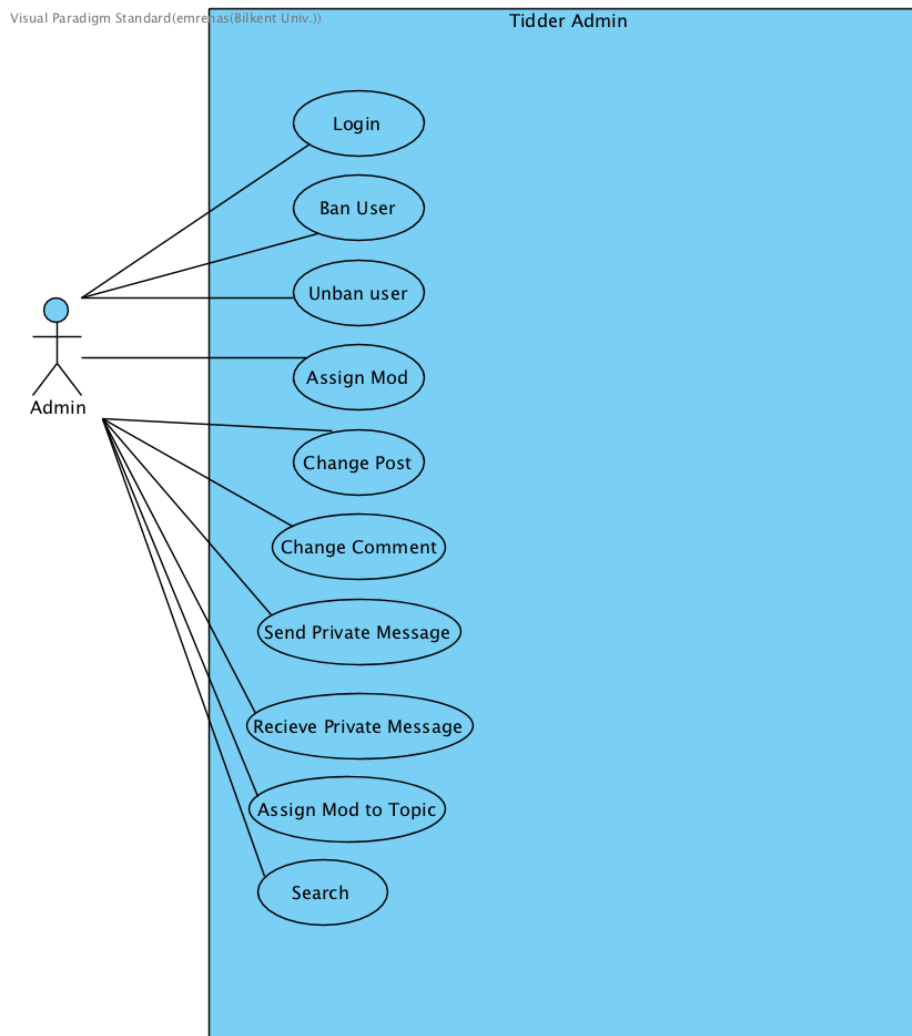
- Login: Admins can login with their username and password

- Ban user: Admin can suspend users to prevent them posting comment or post

- Unban user: Admin can take back ban action to any user

- Change Post: Admin can change any post on any topic

- Chang Comment: Admin can change any comment under any topic

- Send Private Message: User send private message to other users

- Receive Private Message: User receive private messages which sended to them by other users.

- Search: Admin can search anything in the application.

One of the actor of Tidder is admin. Admins are the most authorized actor of the system. They login as same as user. Once they logged they have special page for admins which is admin panel. They can change anybody's post or comment if they find inappropriate. Also if they want they can assign any user to be a mod for any topic they want. They can also use the system as user like creating post or comment or search specific text they seek for.

4.2.3 Mod User
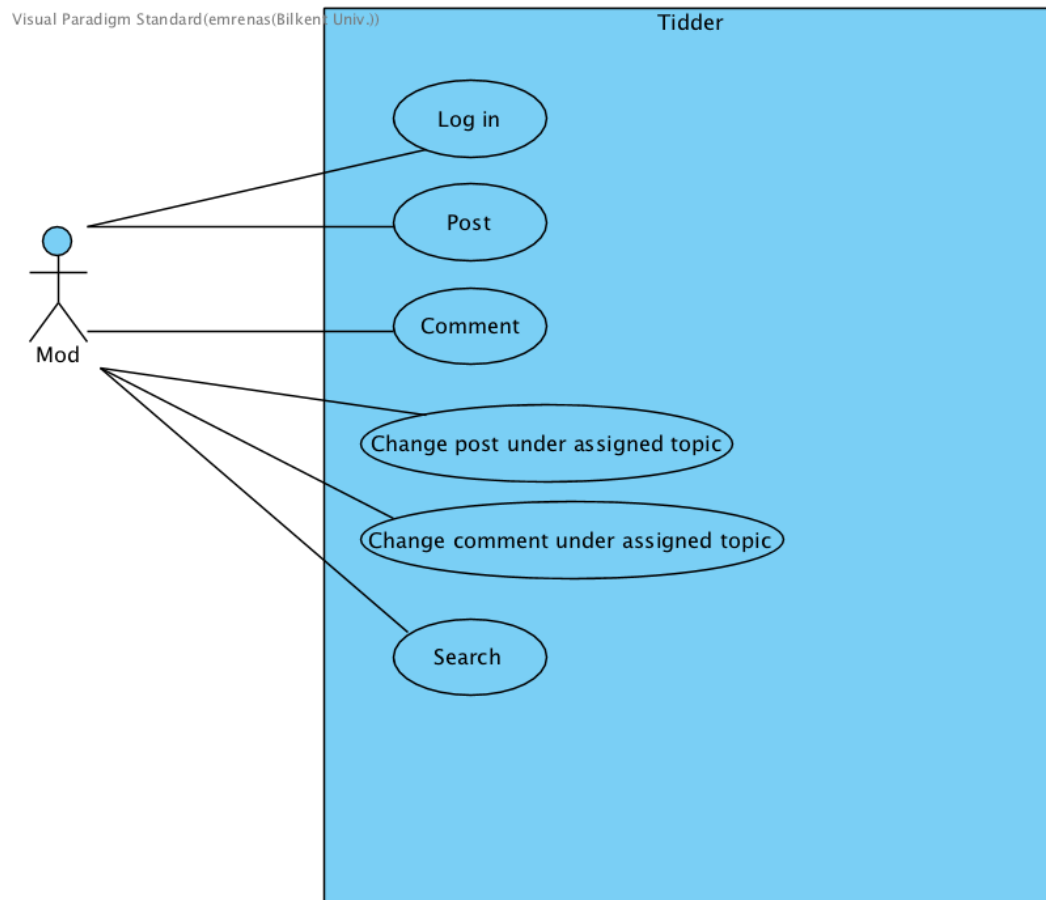
- Login: Mods can login with their username and password

- Change Post under assigned topic: Mods can change/edit any post under their assigned topic

- Chang Comment: Mods can change/edit any comment under the post which they assigned topic

- Send Private Message: Mods send private message to other users

- Receive Private Message: Mods receive private messages which sended to them by other users.

- Search: Admin can search anything in the application.

Mods are assigned by admins and once they assigned they have some privileges in specific topic which they assigned for. They login as any other actor of the system. Once they login

they can use the system same with users. But if they enter into their assigned topics, they can change any comment or post under their assigned toppics.

4.3 Scenarios

4.3.1 Login

Developer Emre wants to login into system. He needs to go to login page and type his username and password. System should compare his username and convert its password to encrypted data and compare username and encrypted password with the database. If the system returns true system will log Emre in into the system.

4.3.1 Create Post

User Umut wants to create a post for mobile phones. After he went to mobile phone topics in the software. He should click create new post button. After he clicked the button system process to create post webpage. He then should type the header of the post and the subject of the post he should click create post button in the webpage. After he clicked system will save the post as activity in the database and send this activity into post database.

4.3.2 Create Comment

User Umut wants to comment a post. He should go to any post that he wants to make a comment. After he is in the page he wants to make comment he should click make a comment. Then he will type the comment and click send comment button. The comment will be send to activity table in database and then the system will take the comment from activity table and send it to comment table.

4.3.3 Search

User Batıkan wants to search a post in the software. He will type what he wants to search in the search field. The system will use sql statement to search in database and show the results to Batıkan.

4.3.4 Change Post

Mod Doğukan wants to change a user's post in his assigned topic. After he logged in he click the change post button and make the necessary changes in the post and he should click to send button. After he clicked system will update the corresponding entry in database table.

4.3.5 Assign Mod

Emre thinks system need a mod in fashion topic. After he decided who should be mod for computer topic. He decides Ece should be mod in fashion topic. Emre needs to go to Ece's profile page and click Assign as Mod button. After he clicked he will choose which topic Ece will be assigned for. After he choose the topic the system will update Ece's profile in database to be a Mod for fashion instead of user.

4.3.6 Ban User

Emre saw a user commented a swear to another user. He thinks this behavior is inappropriate and decide to ban the user. He will go to that user's profile and click ban this user. After he clicked the button, his privileges in the database will be set to 0 where user cannot perform any activity.

4.3.7 Unban User

Emre believes a user is banned unnecessarily so he wants to take user's ban back. He needs to go to user's profile and click Unban button. After he clicked his activity permission's will be set the default.
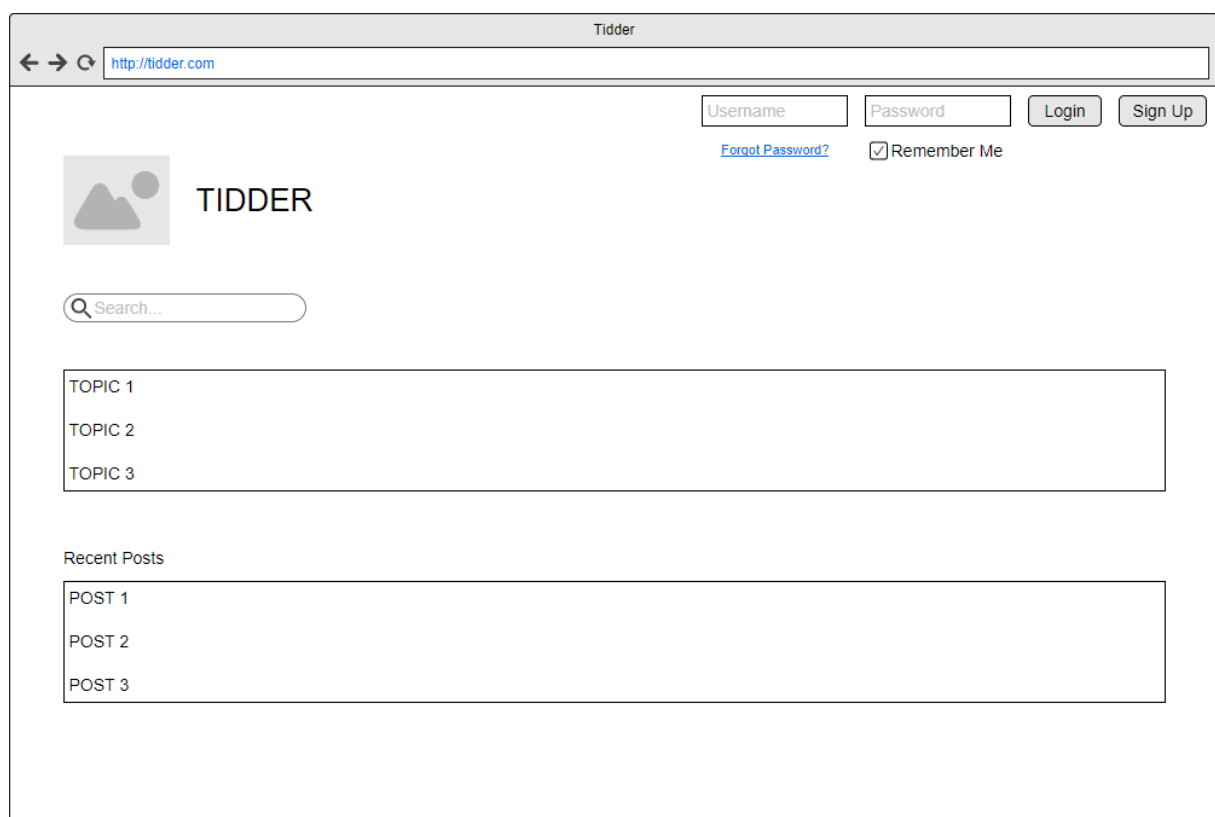
4.3.8 Send Private Message

User Umut want to send a message to user Batıkan that no one else can see this message besides Umut and Batıkan. Umut will go to Batıkan's profile and click Send Message button. After he clicked the button the system will processed to send message page. Umut will type subject and message. After he typed he will click send button. After that system will save the message into message table and Batıkan will receive this message from the same message table.

5 User Interface Design and SQL Statements

5.1 Home Page

Home Page is the first page that a user or a visitor see. The page consists of a login bar, recent posts and each discussion topic grouped in their corresponding areas. The Home Page has a sign-up button in order to redirect unregistered users to the signup page.

Topic area SQL:

SELECT * from TOPIC

      WHERE 1;

Recent Posts SQL:

SELECT TOP 5 * from Post

      ORDER BY post_date DESC

5.2 Home Page Logged In

When a user logged in to the website, the login bar in the home page replaces with some links about user profile link, inbox link,etc.
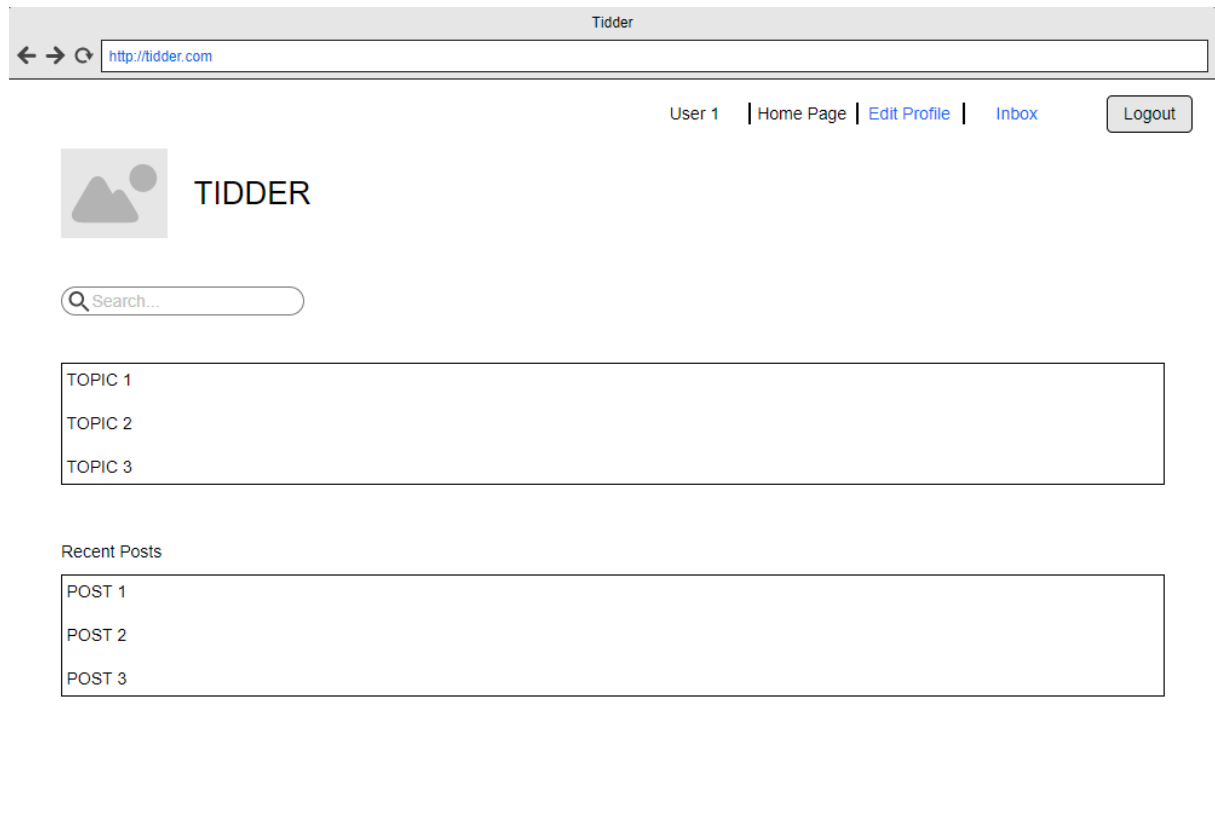
Topic area SQL:
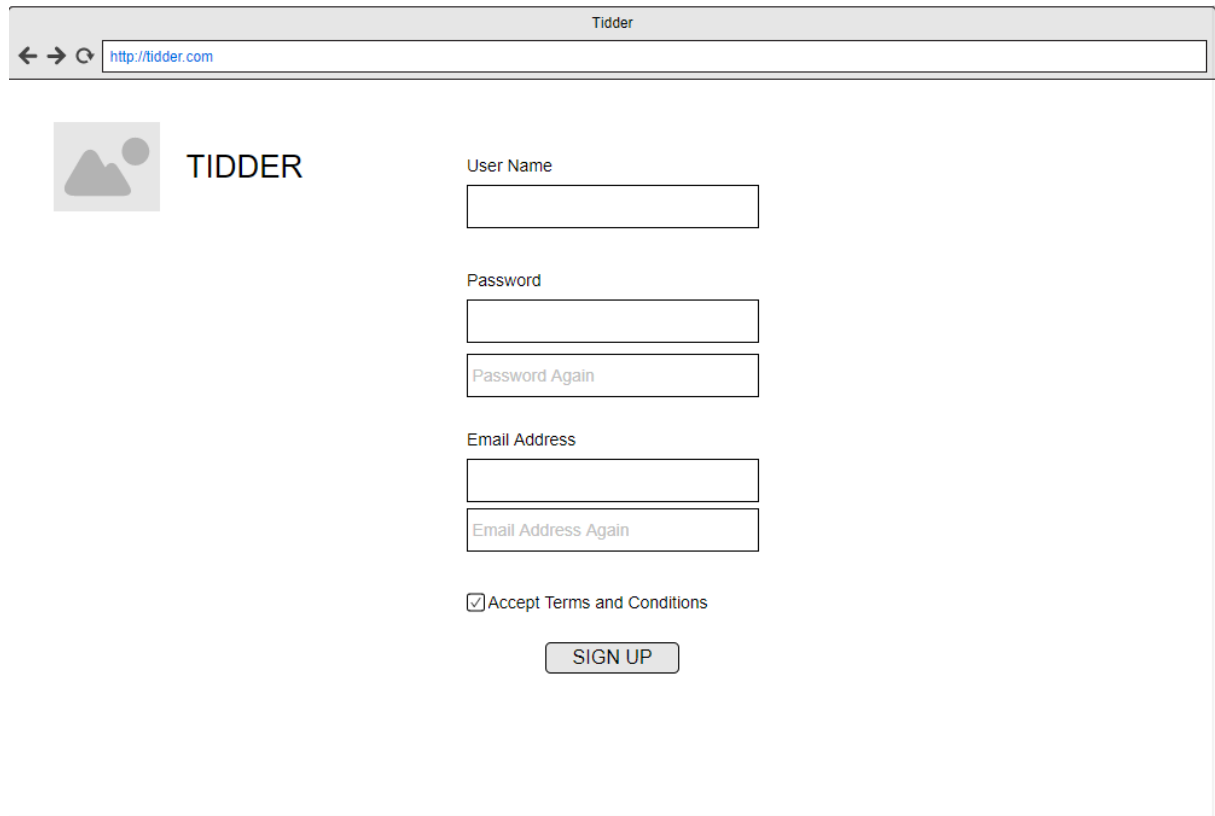
SELECT * from TOPIC

      WHERE 1;

Recent Posts SQL:

SELECT TOP 5 * from Post

      ORDER BY post_date DESC

5.3 Signup Page

Signup page is for creating a new user by entering an email, a username and a password.

After User Clicked SIGN UP Button:

INSERT INTO User (name, password, email,register_date,permissions)

        VALUES(typedName, typedPass, typedEmail,curdate(),userPermission);

5.4 Profile Page

The logged in user can see its profile page through this page. Profile page consists of recent posts of the user, recent activities, its profile picture, its bio and a button for private messaging.
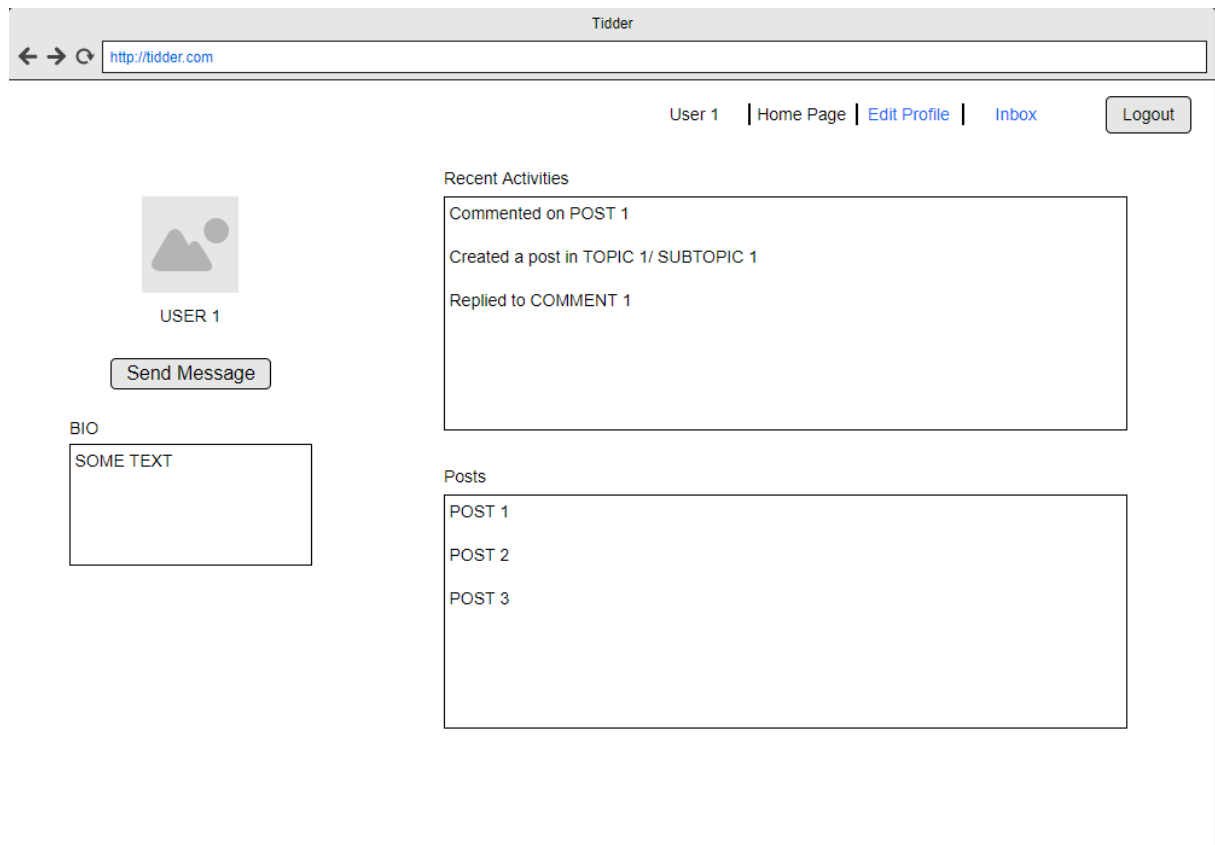
Picture and Bio SQL:

SELECT pic_link, bio, username from User

       WHERE id = '1';

Recent Activities SQL:

SELECT TOP 5 * from Activity_done

       WHERE u_id = '1';

Last 5 Posts that User Posted SQL:

SELECT TOP 5 * from Activity_done, Activity, Activity_applied

       WHERE u_id = id AND type = '3';

5.5 Edit Profile Page

In this page, logged in user can change the information about itself and save it to the database. User can change its bio, name, birth day.

Update User SQL:

UPDATE User

       SET name = typedName, typedSurname, birth_date = changedDate, bio = typedBio

       WHERE id = '1';

5.6 Private Message Page

In Tidder, users can communicate each other using private messages. Each user has an inbox that can see each message send or received. They can select a conversation and type a new message though this page.

Sending Message SQL:

INSERT INTO Message(message_id,text,subject)

       VALUES(message_id+1,typedText,typedSubject)

INSERT INTO mesEvent(sender_id, reciever_id, message_id, date)

       VALUES(userID, typedUser, message_id+1, curdate());

5.7 Topic Page

Users will be redirected to their corresponding topic choice. In this page, users can see recent posts about the topic and subtopic that related to the topic.
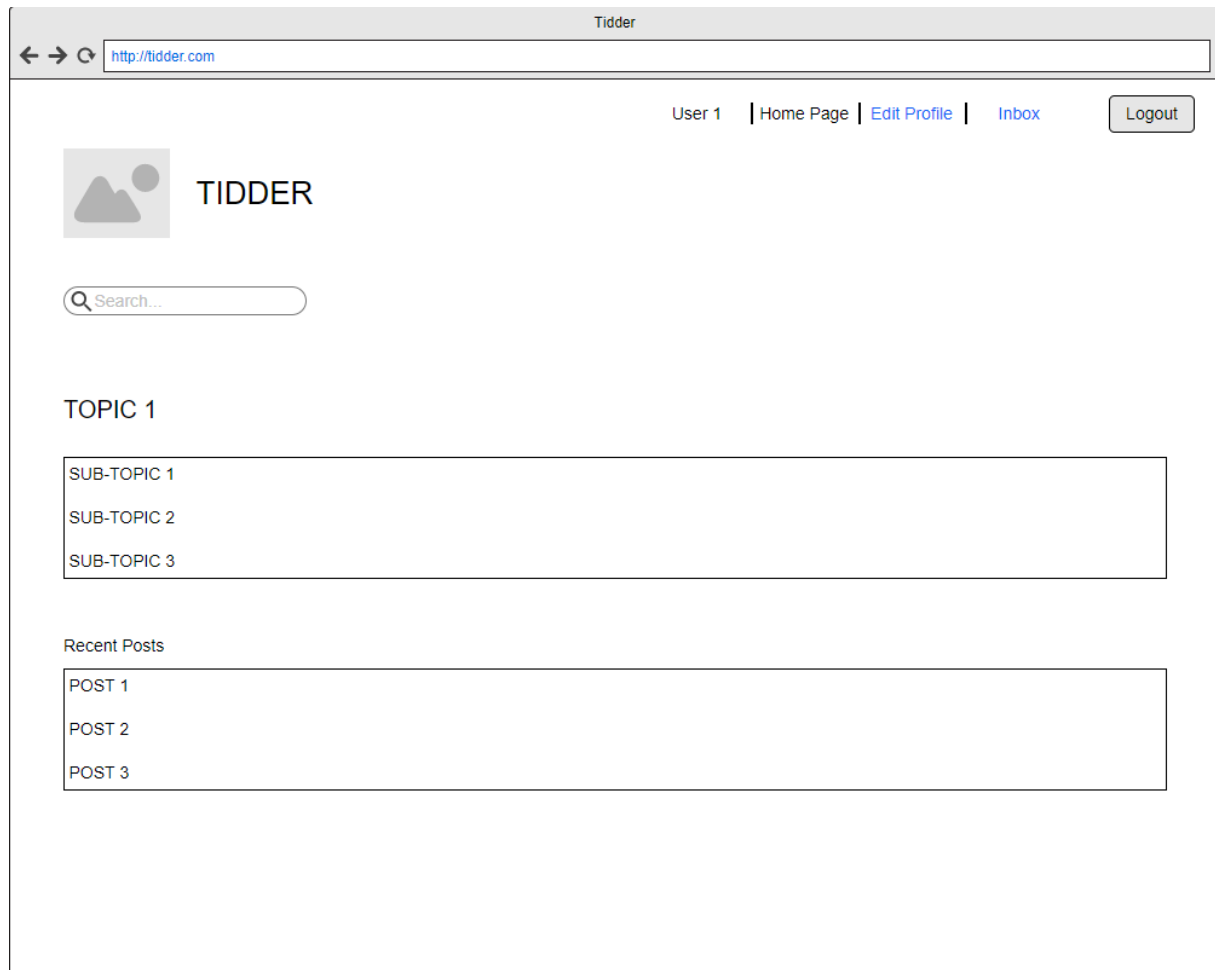
Recent Posts SQL:

SELECT TOP 5 * from Post NATURAL JOIN ContainsPost

      WHERE subtopic_id = subtopicID;

Subtopics SQL:

SELECT * from Subtopic NATURAL JOIN TOPIC

      WHERE topic.id = selectedTopic;

5.8 Subtopic Page

In this page the posts related to the subtopic will be shown and the subtopic's recent posts will be displayed. At this page users can be redirected to the post creation page with the "Add Post" button.

Recent Posts SQL:

SELECT TOP 5 * from Subtopic

    WHERE topic = topicID;

All Posts SQL:

SELECT * from Post NATURAL JOIN ContainsPost

    WHERE (SELECT * from Subtopic

        WHERE topic = topicID);

5.9 Post Creation Page

Users can create posts under the specified subtopic by pressing "Add Post" button. Users will be redirected to this page in order to create the post. Users define a Title for the post and a content text in the text field and simply by clicking "Post" button. Their post will be posted to their desired subtopic.

Adding Post SQL:

INSERT INTO Activity (activity_id, creation_time, type)

    VALUES(activity_id+1,curtime(),(SELECT type as a_type from Activity_type

                          WHERE name = 'add post';))

INSERT INTO Activity_done (u_id,a_id)

    VALUES(userID, activity_id+1)

INSERT INTO Post (post_id, post_user, content)

VALUES(post_id+1,userID,typedContent);

5.10 Post Page

In this page, users can see the content of the post and the comments that have been made to the post. Each comment can have sub comments by each user indefinite times. The users can edit, remove their comments and sub comments and make new comments for the post as well.

Post Content and Comment Retrieval SQL:

SELECT * from Post

       WHERE post_id = 1;

SELECT * from ContainsComment NATURAL JOIN Comment

       WHERE post_id = 1;

SELECT * from Subcomment NATURAL JOIN Comment

       WHERE 1;

Make Comment SQL:

INSERT INTO Activity (activity_id, creation_time, type)

       VALUES(activity_id+1,curtime(),(SELECT type as a_type from Activity_type

                       WHERE name = 'make comment';))

INSERT INTO Comment (comment_id, comment_user,subcomment_count, content)

       VALUES(comment_id+1,userID,0,typedContent);

6 Advanced Database Components

6.1 Constraints

6.1.1 Comment Constraints

Even though users cannot comment without seeing a comment on the webpage, sometimes there can be error in the system if two or more comment send to system at exactly same time the target ID's of comments can interfere with each other, in order to prevent this a constrains will be implemented to check if the comment is exist.

Create assertion comment_constraint check
(exist(select*

      from comment

      where comment_id in (select comment_id

                from subcomment)

6.1.2 Message Constraints

A normal user may try to send private message to banned user. Since banned user are not allowed to do any activity they are also not allowed to make a private conversation with other users, therefore a user must send message to unbanned user and not a banned user.

Create assertion message_constraint checkBan
(select* id
from user
where id=reciever_id and id in (select id

                from user

                where privileges <> 0)

6.2 Triggers

6.2.1 Activity_trigger

When a user perform an activity his/her permissions must be allowed to that action, therefore when new activity occurs activity_trigger should be triggered in order to find out if user is permitted to do that activity.

6.2.2 Moderator_trigger

Since moderators are between a user and admin. They can do what user cannot do in some specific topics. Moderators can change other user's post or comment a trigger must be triggered before executing their action to check if they are changing a comment or post under their assigned topics.

6.3 Views

6.3.1 User Profile View

The view gets the profile information of a user without authentication informations, in order to display the information about a user in the profile page of the website. Therefore a view will be used for this particular operation. This view will also be used in the post page, private message page to retrieve the picture of the users and usernames for each comment and post.

create view user_profile as
select ID, birth_date, bio, pic_link
from User

6.3.2 Recent Posts, Recent Activity View

The view gets the latest 5 post has been posted in each topic/subtopic to display the information in the home page, topic page and subtopic page. Recent Activity view will be used in the profile page in order to retrieve latest activity of the user to display.

```
create view recent_subtopic_posts as

select top 5 * from Post natural join ContainsPost

where subtopic_id = subtopicID;

create view recent_topic_posts as

select top 5 * from Post natural join ContainsPost

where subtopic_id = subtopicID;

create view recent_user_activity as

select top 5 * from Activity_done

where u_id = '1'
```

6.4 Reports

6.4.1 The Most Active User

The query returns the most active user in terms of activity done:

```
create view as active_users

select  *

from   activity_done natural join activity

group by u_id

order by count(*) desc

limit   1;

select top 1 * from active_users
```

6.5 Stored Procedures

Tidder is an application that anyone can discuss about anything, therefore, it will be grow as much as it needs in terms of topics. Admins can add new topics to the system but before, a new topic is created a moderator for that topic must be decided in order to control and keep discussions within the frame of respect. Therefore, a procedure will be implemented for creating a topic and assign a moderator to that topic at the same time.

7 Implementation Plan

We are planning to use MySQL for database management system. For back-end applications, we are eager to use Python 2.7 with Django Framework. Although this framework provide advanced query systems, we will still use legacy SQL queries. For front-end applications, we will use Bootstrap, Javascript, HTML and CSS where needed.