

**Database Systems**

**CS353**

**Final Report**

**Group 6**

**Ahmet Emre Nas 21402357**

**Doğukan Altay 21400627**

**Umut Akös 21202015**

**Batıkan Hayta 21301382**

[**1. Brief Description**](#_v83qtyxy7cnc) **3**

[**2. Final E/R Diagram**](#_j7sesdu4y3ug) **4**

[**3. Relation Shema**](#_idtfbsamjp04) **5**

[**3.1.1 User**](#_besbxbooh8zy) **5**

[**3.1.2 Message**](#_ktyyhwze6t6d) **5**

[**3.1.3 mesEvent**](#_xuinl1bcggsv) **5**

[**3.1.4 Assigned**](#_o6xi8wh1fdoo) **5**

[**3.1.5 Activity**](#_m0alumkmlvse) **6**

[**3.1.6 Activity\_type**](#_fheozf4s0hd8) **6**

[**3.1.7 Topic**](#_te7ooaaauu7a) **6**

[**3.1.8 Subtopic**](#_ktkwrbrboljh) **6**

[**3.1.9 Post**](#_g4fghkkam0xq) **7**

[**3.1.10 Comment**](#_tna0vcgmcoa8) **7**

[**3.1.11 ContainsPost**](#_pt4n5ynw4xfc) **7**

[**3.1.12 ContainsComment**](#_6wlqq4equ826) **7**

[**3.1.13 subComment**](#_ngw0vc8xzlgp) **7**

[**4 Implementation Details**](#_9omg46o5hkrg) **8**

[**6 Advanced Database Components**](#_piu1fw6xjgfk) **9**

[6.1 Constraints](#_dav9kq8m20a6) 9

[**6.1.1 Comment Constraints**](#_hi710jl4217r) **9**

[**6.1.2 Message Constraints**](#_9xcfvgmnk0hv) **9**

[**6.2 Triggers**](#_7a809ox5gyo8) **10**

[6.2.1 Activity\_trigger](#_c5h56ogko2wt) 10

[6.2.2 Moderator\_trigger](#_4k8176og7dv6) 10

[**6.3 Views**](#_4o2fw44ce2a6) **10**

[6.3.1 User Profile View](#_rhn46uyyxyn6) 10

[6.3.2 Recent Posts, Recent Activity View](#_xkp45tby6cak) 11

[**6.4 Reports**](#_hr1ih52ua1ht) **11**

[6.4.1 The Most Active User](#_8kkb71m3yfw3) 11

[**6.4 User Manual**](#_pa1w03139mhe) **12**

# 

# 1. Brief Description

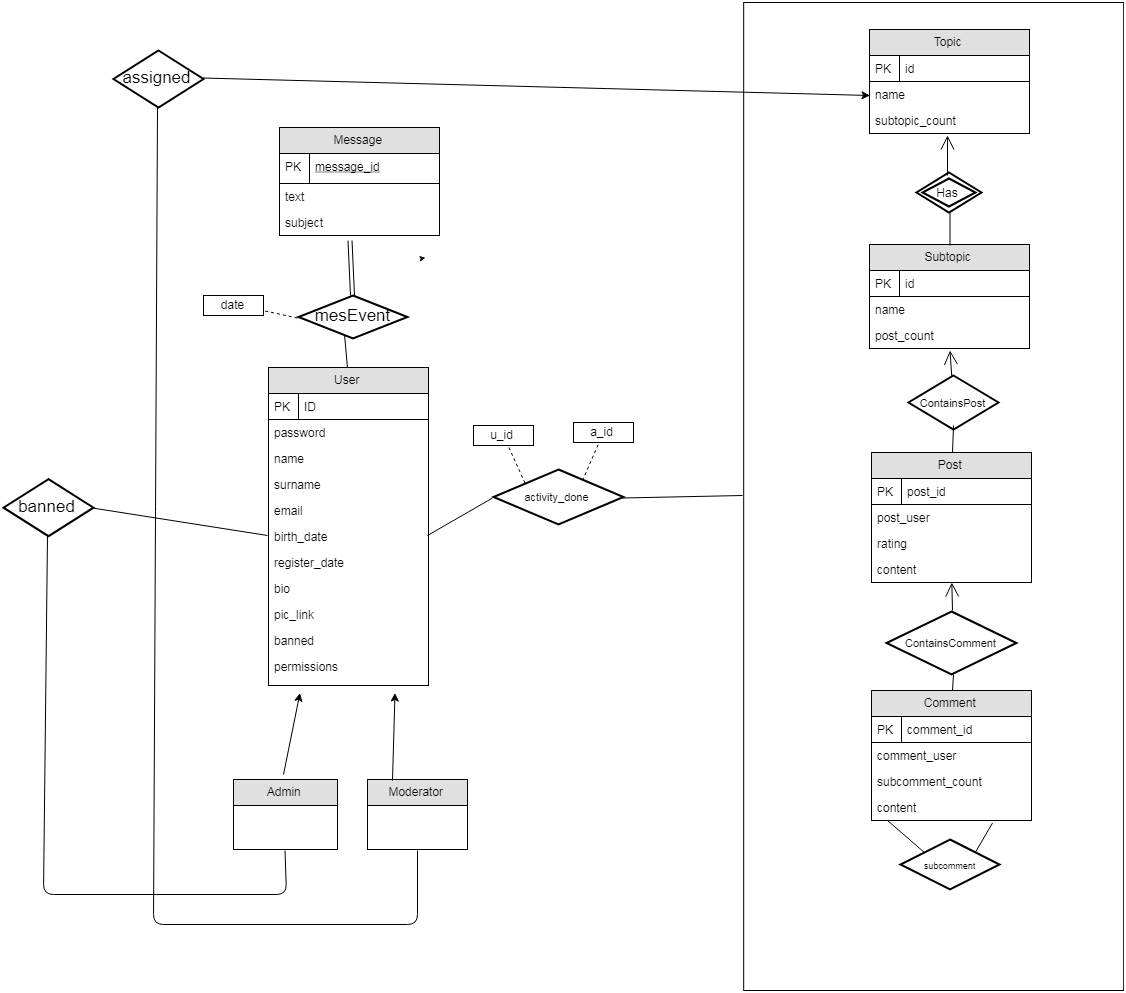
Our application Tidder aims to connect every person all around word to discuss any topic a person can think about. A user can create any topic about a question a, discussion or a news. Other users of the application can comment to these topics. Even though these topics are public and visible to anyone, 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. Final E/R Diagram

****

# 

# 

# 3. Relation Shema

# 3.1.1 User

Relational Model

**user(**ID, password, name, email, birth\_date, register\_date, bio , pic\_link,username, surname)

Primary Key -> ID

Candidate Key -> ID

# 3.1.2 Message

Relational Model

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

Primary Key -> ID

Candidate Key -> ID

# 3.1.3 mesEvent

Relational Model

**mesEvent(**message\_id, sender\_id, receiver\_id, date)

foreign key message\_id references message(id)

foreign key sender\_id references user(id)

foreign key receiver\_id references user(id)

# 3.1.4 Assigned

Relational Model

**assigned(**mod\_id, topic\_id)

foreign key mod \_id references mod(id)

foreign key topic \_id references topic(id)

# 3.1.5 Activity

Relational Model

**activity(**activity\_id, creation\_time, type, content)

PRIMARY KEY -> activity\_id

CANDIDATE KEY -> activity\_id

# 3.1.6 Activity\_type

Relational Model

**Activity\_type(**name, type)

PRIMARY KEY -> name

CANDIDATE KEY -> name

# 3.1.7 Topic

Relational Model

**topic(**id, name, definition)

PRIMARY KEY -> id

CANDIDATE KEY -> id

# 3.1.8 Subtopic

Relational Model

**Sub\_topic(**id, name, definition)

PRIMARY KEY -> id

CANDIDATE KEY -> id

# 3.1.9 Post

Relational Model

**post (**post**\_**id, u\_id,content,date,time,subject)

PRIMARY KEY -> post\_id

CANDIDATE KEY -> post\_id

# 3.1.10 Comment

Relational Model

**comment (**comment**\_**id,u\_id,content)

PRIMARY KEY -> comment\_id

CANDIDATE KEY -> comment\_id

# 3.1.11 ContainsPost

Relational Model

**containsPost (**subtopic\_id, post\_id,subtopic)

FOREIGN KEY -> subtopic\_id references subtopic(id)

FOREIGN KEY -> post\_id references post(id)

# 3.1.12 ContainsComment

Relational Model

**containsComment (**id,comment\_id, post\_id,comment\_id,user\_id,date,time)

PRIMARY KEY -> id

FOREIGN KEY -> comment\_id references comment(id)

FOREIGN KEY -> post\_id references post(id)

# 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)

# 4 Implementation Details

**FRONT-END**

The front-end of the TIDDER web app is run using the HTML, CSS, JavaScript and Bootstrap. Bootstrap allow us to design simple user interfaces for users. HTML is the standard programming language for describing the contents and appearance of Web pages and by the help of the CSS, it makes the web sites more presentable. For interaction between users and web-site, we use JavaScript. It runs on the user’s computer and does not require a downloads from the website.

**BACK-END**

In TIDDER back-end web app, we used Python 2.7 with Django Framework. Although this framework provide advanced query systems, we used legacy SQL queries. All the tables and queries coded in python. Passing the variable in python to html, we used Jinja which comes with Django Framework. In html files, users cannot see the Jinja codes however, it provides us an easy usage in html files.

**DATABASE**

TIDDER uses MySQL for data storage and querying. All the queries processed in MySQL database server and stored in MySQL database. Python code send the queries to MySQL database and those queries are proceed by the server. Also, in query processing, we are not allow any user to escape a special characters to use any SQL injection to our database to steal information to our database.

Django frameworks is a MVC design pattern and it is different than other web-application because most of the web applications use server-client based design pattern. It was hard to figure it out the adjust this pattern to our web-site but later on it help us the do back-end and front-end easily. Unlike the php, you need to divide each part for views, models, forms etc. and use them accordingly to your need.

# 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 Register Date Trigger**

When a new user signed up this trigger will initialize in order to decide the registration date of the user.

CREATE TRIGGER registerUserTime

ON User

AFTER INSERT

AS BEGIN

UPDATE User

SET registration\_date = GETDATE()

FROM User JOIN INSERTED i

WHERE i.ID = User.ID

END

## 

## **6.2.2 Post Date Trigger**

When a new post created this trigger will initialize in order to decide the creation time of the post.

CREATE TRIGGER postCreationTime

ON Post

AFTER INSERT

AS BEGIN

UPDATE User

SET registration\_date = GETDATE()

FROM User JOIN INSERTED i

WHERE i.ID = User.ID

END

# **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

order by date, time desc

where subtopic\_id = subtopicID;

create view recent\_topic\_posts as

select top 5 \* from Post natural join ContainsPost

order by date, time desc

where topic\_id = topicID;

create view recent\_user\_activity as

select top 5 \* from Activity\_done

order by time desc

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 \*

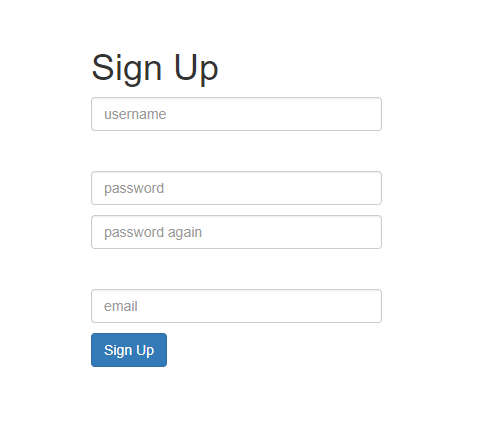
from activity\_done

group by u\_id

order by count(\*) desc

limit 1;

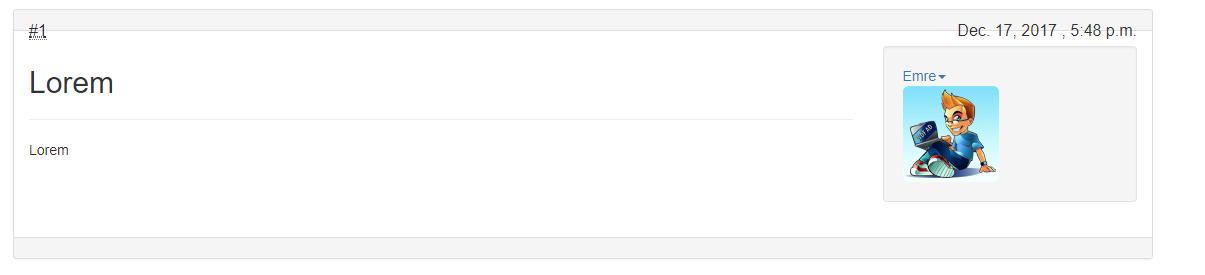
# 6.4 User Manual



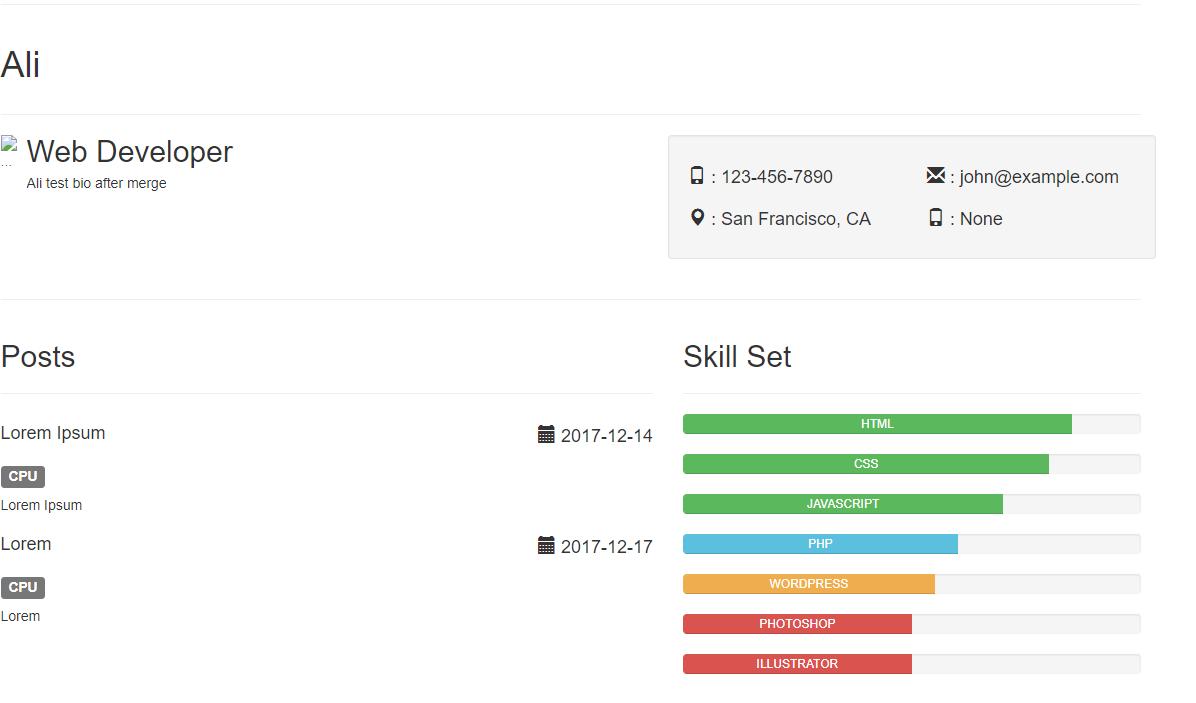
* If user wants to login into system, he needs to go to login page and type his username and password and the system returns true system will log user in into the system.
* If user wants to sign up the TIDDER, he should click the sign up button and he should fill username, password and his mail address. If user write wrong mail or short and unnecessary password, system does not allow the user and wants them again from his.

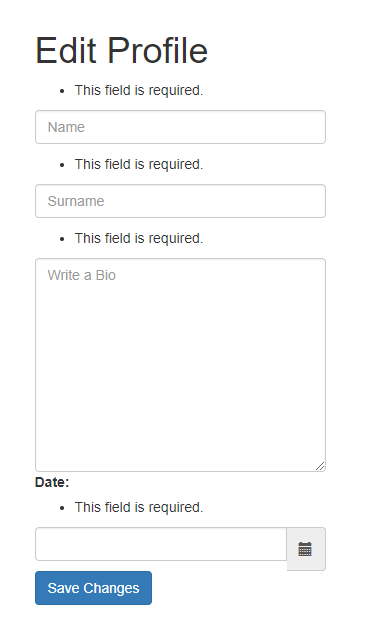


* After user login the TIDDER, he can see left-up of the page Home, Messages, Edit Profile and See Profile.



* If he clicks the Home button, he can see topics and all topics has subtopic. Under the topics, there is some definitions what topic is about and user clicks a topic, he can see that some topics about main topic. Ex: Computers → CPU. In there Computers is under control of the Admin so only admin can change, create and delete topics but CPU is for user and they can post to this subtopic. Also there are other stuff, An Unordered List and An Ordered List in the homepage.
* In subtopic page, user can post and see other post also if he clicks the other users he can see the profile of them, send PM, view all his articles, view all his posts, add user hic contact list. Also there is a manage user button, but it is only for admin.
* In message button, user can see all messages and left of the page, he can select one of them and send message to another user.





* Edit profile page, user can change his profile, to do that he should write his name, surname, his biography also birth date.
* In all pages in the TIDDER, a search button seems on right of the page and user can search whatever he wants about TIDDER.