|  |
| --- |
| C:\Users\Administere\AppData\Local\Microsoft\Windows\INetCache\Content.Word\mvp-1.png  *ShareBoard is our little creation where we try to bring together different features from social networking platforms like Facebook, Twitter and Google+ to create a stable, user-friendly and unique social media platform.* |
| ShareBoard |
| DBMS Laboratory Project |
|  |
| **Sanghamesh (41) and Sriharsha (50)** |
| **15/11/2016** |

**Submitted to,**

DIVAKARA  .N   
Assistant Professor,  
Department of Computer Science and Engineering,  
Sri Jayachamarajendra College of Engineering(SJCE).

**Contents**

**1. Introduction**

* 1. **Objective of the project**

To create a social media application which utilizes a database to store all the information about the user as well as the content shared on the social media platform. The use of DBMS and related areas like creation, update and delete of tables, triggers, views, etc is given more priority and seen to it that the project is completed within the limits of the constraints provided to us. We try to create an application similar to modern social media platforms like Facebook and twitter but has its own unique features which helps it stand out from the rest.

* 1. **Features of the project**

ShareBoard, as the name suggests lets users to share their thoughts and ideas on a platform intended to bring together users from different walks of life. From Politics to Education, Sports to Travel, we try to cover all the topics an user might be interested in. Like all web applications, an user will have to first register in order to use the application by providing details like his Name, DOB, gender and a unique username.

Security is provided through password for further login sessions for a particular username. He can select a list of topics pertaining to his interests which will determine the post he sees when logs in. Once registered, he can login to view others shares, search for shares using hashtags, see his account activities, edit preferences and settings and a lot more.

Shares are basically mini posts with a title, body, name of the publisher and timestamp. It contains reactions or comments posted by other users. A share also consists of hashtags; so any post containing a particular hashtag can be listed. Shares are categorized into predetermined types such as “Sports”, “Politics”, “Education”, “Travel”, “Technology”, etc.

A Comment is user opinion on a specific share and contains details such as comment date, username. While Reactions are a set of predetermined emotions that can be expressed by a user against a particular share, comments can be any textual opinion written by the user.

**To summarize, the features provided are as follows:**

* New user registration.
* Existing user login.
* Creation of shares.
* Listing of shares.
* Reacting to shares.
* Opinions about a share in form comments.
* Grouping of shares with respect to type of share.
* Searching posts containing specified hashtags.
* Look into his account details.
* Edit preferences using ‘Settings’.

**2. System Requirement Specification (SRS)**

**2.1 Intended Audience and Product Scope**

The intended audience for our project includes a common person looking for news to someone who wants to share their ideas with the world. The type configuration features enable a user to look for precise topics he/she is interested in. Not just that, anyone can put up an idea open for funding; other users will be directed to the product site directly through links. In a nutshell, it can serve as a great platform for like minded people to come together according to their interests.

Also, a practical use can be in any organization where employees can come together to discuss about the ongoing projects in the company. We can group posts according to projects and hence, our application can provide a framework for team communication within the company which otherwise is very cumbersome through mails! Also, the company can keep track of all of its projects in a much easier and organized way.

**2.2 Overall Description**

As mentioned earlier, the web based application must enable users to login, post, comment and react to posts. All this is achieved using backend technologies like JDBC and JSP. For the front end, we use HTML, CSS, JQuery and AJAX. All these technologies need to co-operate and communicate with each other in order to provide best possible experience to the user. Authentication and Authorization is given at most importance for the privacy and security of user.

**2.3 External Interface Requirements**

**User Interface**

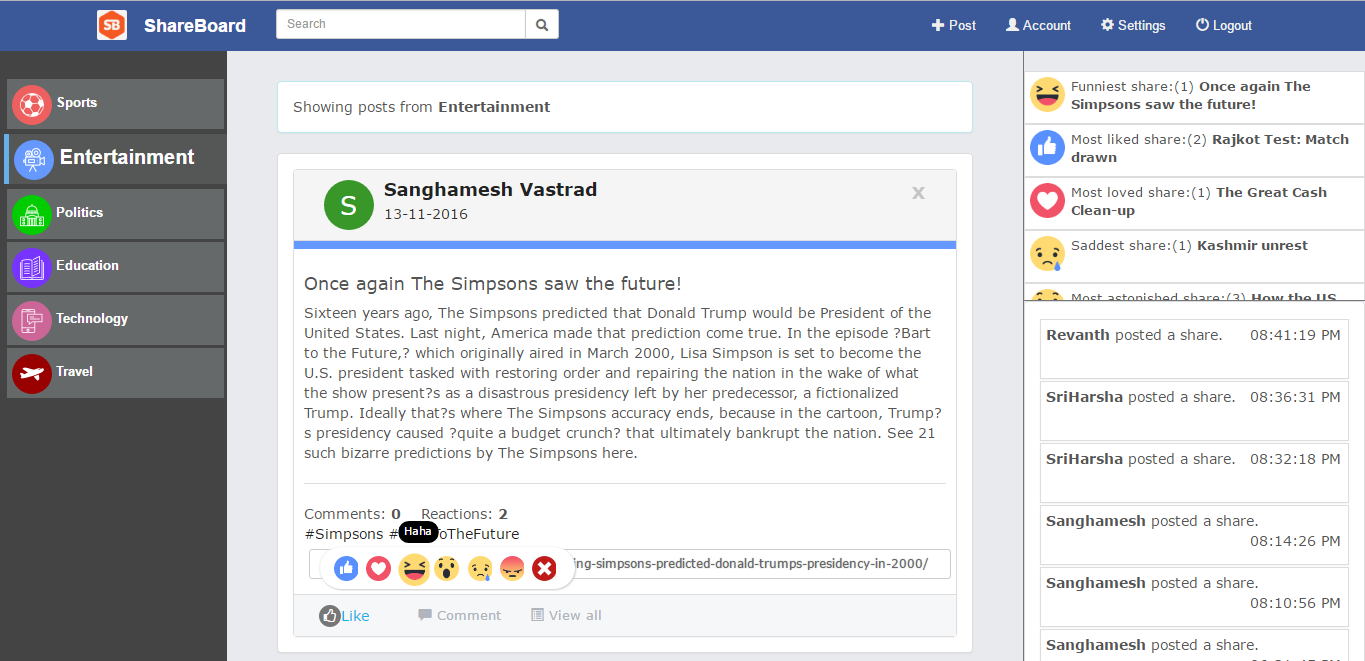
The user interface is designed using HTML and CSS. We have taken care about the design such that there are minimal surprises to the user. The user interface is user-friendly and designed with a natural flow. Also it is taken care that the user has less to enter and remember thanks to Google’s auto complete feature on the browsers. The interface works in full screen mode which gives the user in-depth experience and feel. Colours are used efficiently with integration to facilities; posts of a particular topic will have a particular colour.

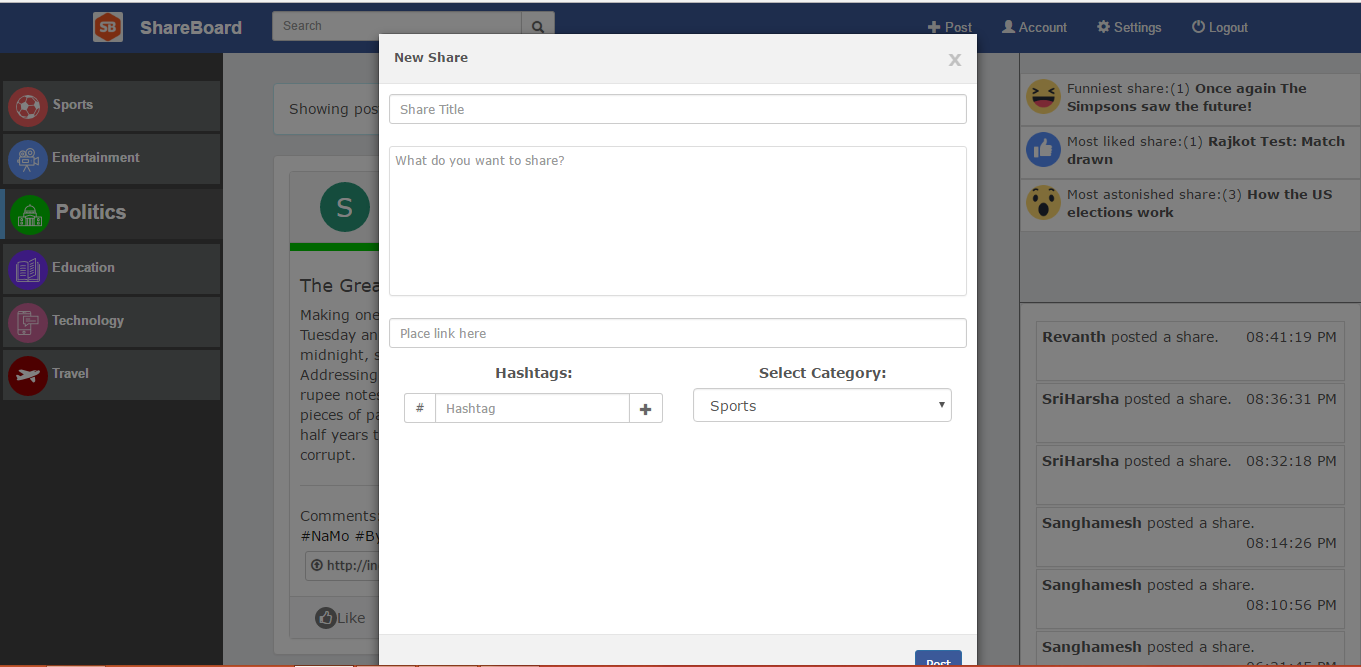
*Please refer to next page for some snippets of our UI.*

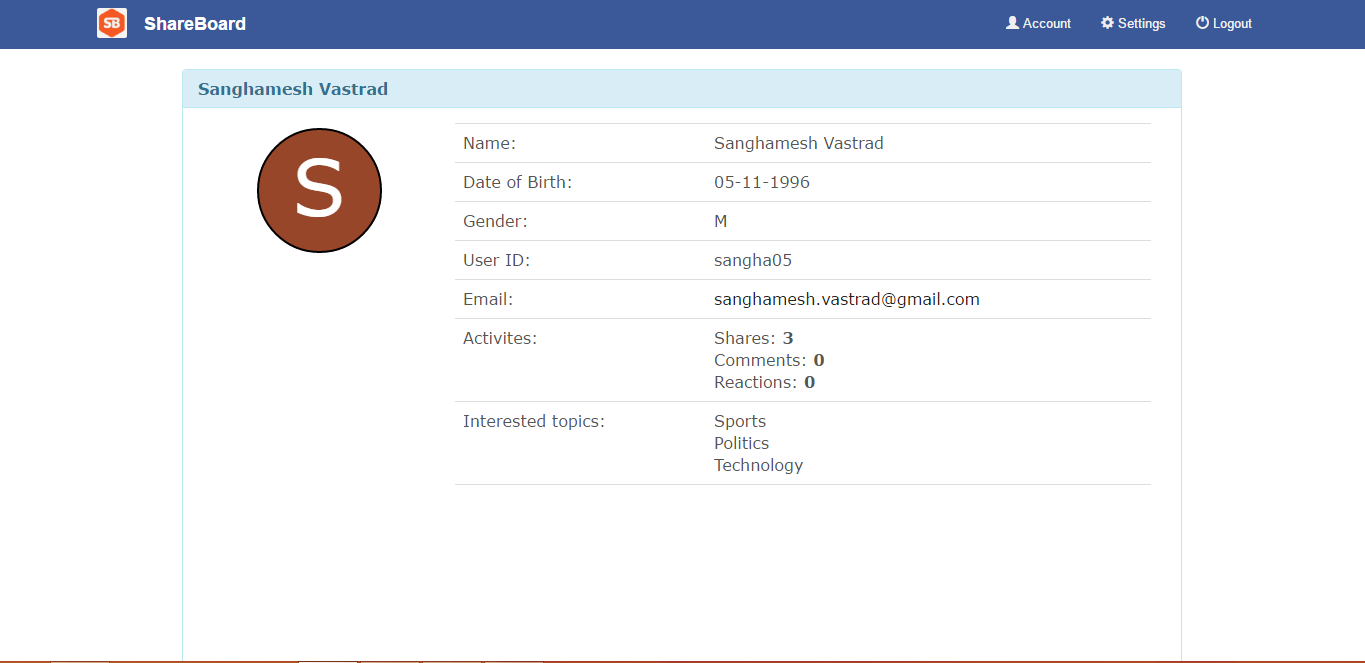
**Communication Interface**

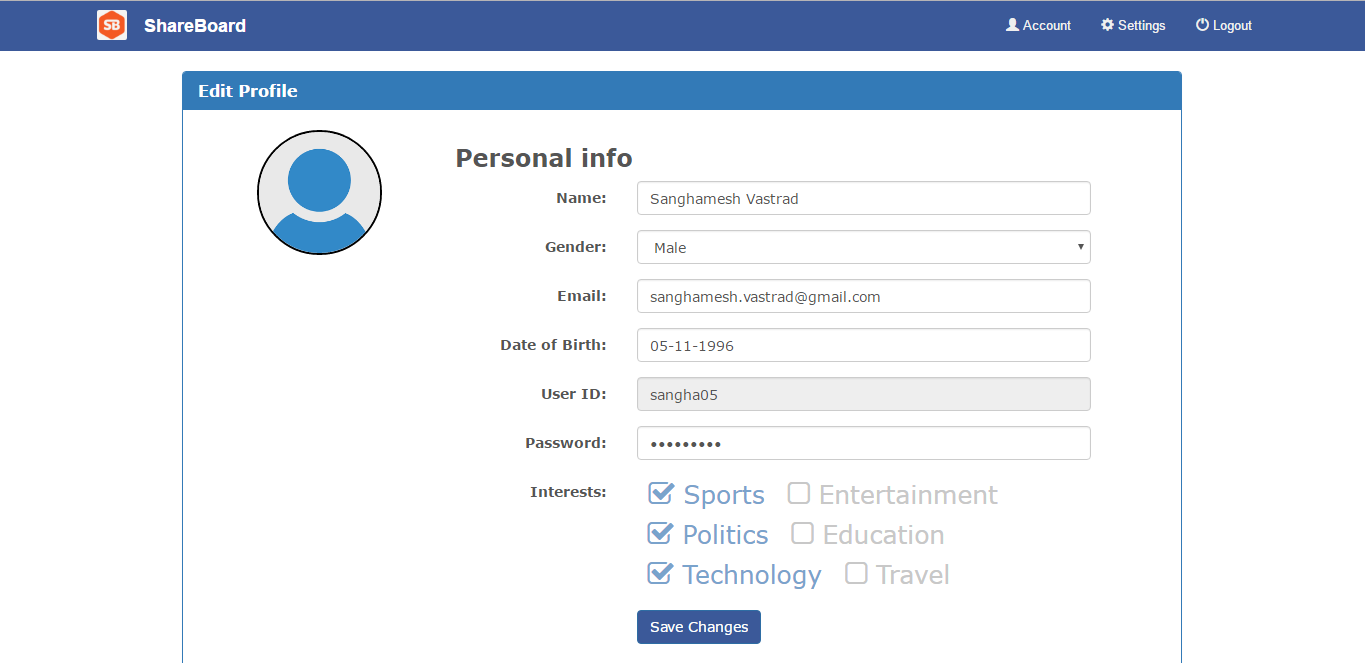
Communication between the database and JSP takes place using JDBC; this is mirrored to the front end through JSP and HTML itself. We will require a server for the web application to work, for which we have chosen Apache Tomcat. For storing the database, we have made use of XAMPP, which provides mysql interface.







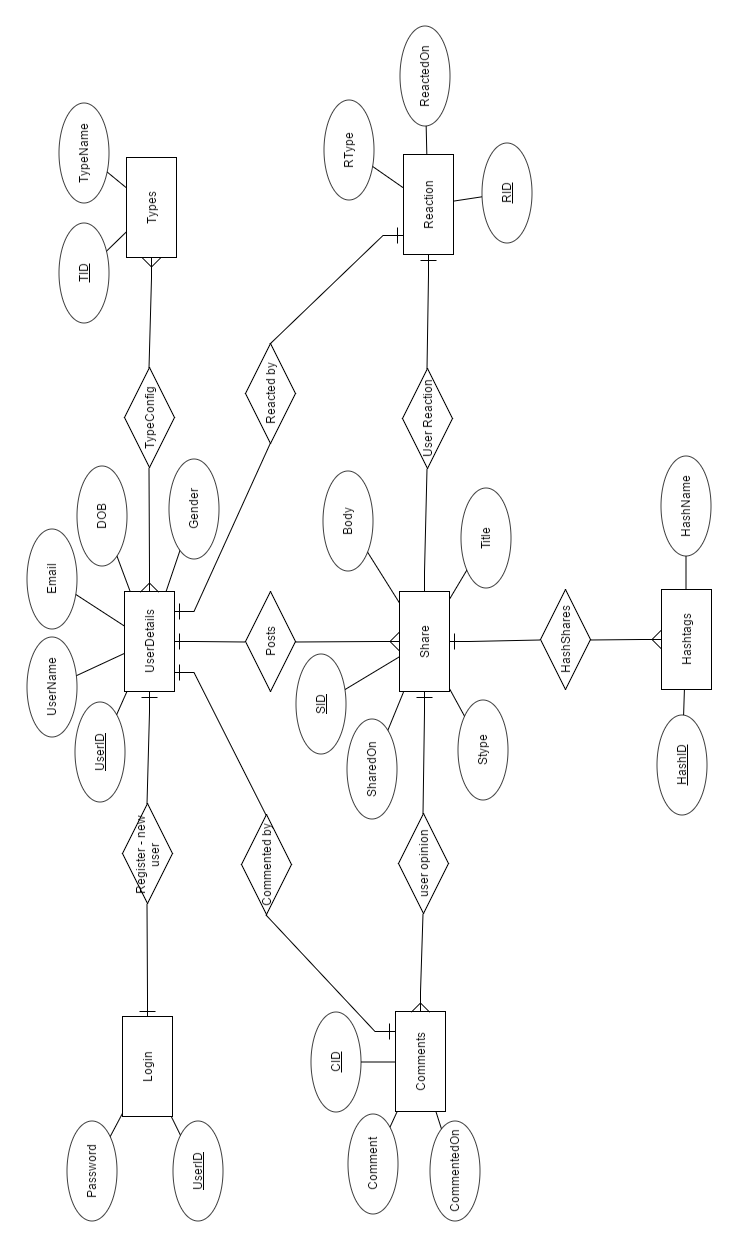




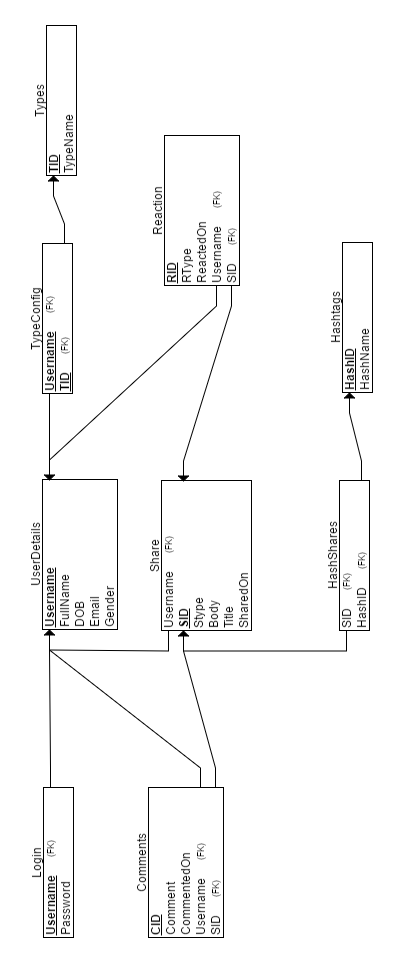


**3. System Design**

**3.1 ER Diagram**

****

**3.2 Relation Schema Diagram**

****

**4. System Implementation**

**4.1 Table, Triggers and Views Creation**

--

-- Database: `sb`

--

--

-- Procedures

--

CREATE DEFINER=`root`@`localhost` PROCEDURE `getIndCmntCount`(IN `Shareid` INT, OUT `count` INT)

READS SQL DATA

SELECT COUNT(\*) INTO count FROM `comments` WHERE Sid=Shareid;

CREATE DEFINER=`root`@`localhost` PROCEDURE `getIndReactCount`(IN `Shareid` INT, OUT `count` INT)

READS SQL DATA

SELECT COUNT(\*) INTO count FROM `reactions` WHERE Sid=Shareid;

CREATE DEFINER=`root`@`localhost` PROCEDURE `getUserCmnts`(IN `user` VARCHAR(255), OUT `count` INT)

READS SQL DATA

SELECT COUNT(\*) INTO count FROM comments WHERE username=user;

CREATE DEFINER=`root`@`localhost` PROCEDURE `getUserReacts`(IN `user` VARCHAR(255), OUT `count` INT)

READS SQL DATA

SELECT COUNT(\*) INTO count FROM reactions WHERE username=user;

CREATE DEFINER=`root`@`localhost` PROCEDURE `getUserShares`(IN `user` VARCHAR(255), OUT `count` INT)

READS SQL DATA

SELECT count(\*) INTO count FROM shares WHERE username=user;

-- --------------------------------------------------------

--

-- Table structure for table `commentnotif`

--

CREATE TABLE IF NOT EXISTS `commentnotif` (

`Sharer` varchar(255) DEFAULT NULL,

`Commtr` varchar(255) DEFAULT NULL,

`cmntOn` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--

-- Table structure for table `comments`

--

CREATE TABLE IF NOT EXISTS `comments` (

`CId` int(11) NOT NULL AUTO\_INCREMENT,

`SId` int(11) DEFAULT NULL,

`Comment` text,

`CmntOn` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`username` varchar(255) DEFAULT NULL,

PRIMARY KEY (`CId`),

KEY `SId` (`SId`),

KEY `username` (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=10 ;

--

-- Triggers `comments`

--

DROP TRIGGER IF EXISTS `onComment`;

CREATE TRIGGER `onComment` AFTER INSERT ON `comments`

FOR EACH ROW BEGIN

INSERT INTO commentnotif

SELECT U.fullname as Sharer,U2.fullname as Cmntr,C.cmntOn FROM comments C,userdetails U,userdetails U2 WHERE C.username = U2.username AND C.Sid = (SELECT ShareId FROM shares S WHERE S.ShareId=C.SId AND username=U.username);

END;

-- --------------------------------------------------------

--

-- Table structure for table `hashshares`

--

CREATE TABLE IF NOT EXISTS `hashshares` (

`hashid` int(11) NOT NULL,

`Shareid` int(11) NOT NULL,

PRIMARY KEY (`hashid`,`Shareid`),

KEY `Shareid` (`Shareid`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--

-- Table structure for table `hashtag`

--

CREATE TABLE IF NOT EXISTS `hashtag` (

`hashid` int(11) NOT NULL AUTO\_INCREMENT,

`hashName` varchar(100) DEFAULT NULL,

PRIMARY KEY (`hashid`),

UNIQUE KEY `hashName` (`hashName`),

UNIQUE KEY `hashName\_2` (`hashName`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=31 ;

-- --------------------------------------------------------

--

-- Table structure for table `login`

--

CREATE TABLE IF NOT EXISTS `login` (

`username` varchar(255) NOT NULL,

`pass` varchar(50) DEFAULT NULL,

PRIMARY KEY (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--

-- Table structure for table `postnotif`

--

CREATE TABLE IF NOT EXISTS `postnotif` (

`username` varchar(255) DEFAULT NULL,

`sharedon` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--

-- Table structure for table `reactions`

--

CREATE TABLE IF NOT EXISTS `reactions` (

`RId` int(11) NOT NULL AUTO\_INCREMENT,

`RType` varchar(10) DEFAULT NULL,

`Sid` int(11) DEFAULT NULL,

`RctOn` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

`username` varchar(255) DEFAULT NULL,

PRIMARY KEY (`RId`),

KEY `Sid` (`Sid`),

KEY `username` (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=12 ;

-- --------------------------------------------------------

--

-- Table structure for table `shares`

--

CREATE TABLE IF NOT EXISTS `shares` (

`ShareId` int(11) NOT NULL AUTO\_INCREMENT,

`SType` int(11) DEFAULT NULL,

`STitle` text,

`SBody` text,

`Slink` text,

`username` varchar(255) DEFAULT NULL,

`SharedOn` timestamp NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,

PRIMARY KEY (`ShareId`),

KEY `SType` (`SType`),

KEY `username` (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=24 ;

--

-- Triggers `shares`

--

DROP TRIGGER IF EXISTS `postnotif`;

CREATE TRIGGER `postnotif` AFTER INSERT ON `shares`

FOR EACH ROW BEGIN

INSERT INTO `postnotif` VALUES (NEW.username,NEW.SharedOn);

END;

-- --------------------------------------------------------

--

-- Table structure for table `types`

--

CREATE TABLE IF NOT EXISTS `types` (

`tid` int(11) NOT NULL AUTO\_INCREMENT,

`tname` varchar(50) DEFAULT NULL,

PRIMARY KEY (`tid`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=7 ;

-- --------------------------------------------------------

--

-- Table structure for table `typesconfig`

--

CREATE TABLE IF NOT EXISTS `typesconfig` (

`username` varchar(255) NOT NULL,

`tid` int(11) NOT NULL,

PRIMARY KEY (`tid`,`username`),

KEY `username` (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--

-- Table structure for table `userdetails`

--

CREATE TABLE IF NOT EXISTS `userdetails` (

`fullname` varchar(255) NOT NULL,

`username` varchar(255) NOT NULL,

`email` varchar(255) DEFAULT NULL,

`dob` date DEFAULT NULL,

`gender` char(1) DEFAULT NULL,

PRIMARY KEY (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- --------------------------------------------------------

--

-- Stand-in structure for view `usermindets`

--

CREATE TABLE IF NOT EXISTS `usermindets` (

`fullname` varchar(255)

,`username` varchar(255)

,`email` varchar(255)

);

-- --------------------------------------------------------

--

-- Structure for view `usermindets`

--

DROP TABLE IF EXISTS `usermindets`;

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `usermindets` AS select `userdetails`.`fullname` AS `fullname`,`userdetails`.`username` AS `username`,`userdetails`.`email` AS `email` from `userdetails`;

-- --------------------------------------------------------

--

-- Constraints for dumped tables

--

--

-- Constraints for table `comments`

--

ALTER TABLE `comments`

ADD CONSTRAINT `comments\_ibfk\_1` FOREIGN KEY (`SId`) REFERENCES `shares` (`ShareId`) ON DELETE CASCADE,

ADD CONSTRAINT `comments\_ibfk\_2` FOREIGN KEY (`username`) REFERENCES `userdetails` (`username`);

--

-- Constraints for table `hashshares`

--

ALTER TABLE `hashshares`

ADD CONSTRAINT `hashshares\_ibfk\_2` FOREIGN KEY (`Shareid`) REFERENCES `shares` (`ShareId`) ON DELETE CASCADE,

ADD CONSTRAINT `hashshares\_ibfk\_1` FOREIGN KEY (`hashid`) REFERENCES `hashtag` (`hashid`) ON DELETE CASCADE;

--

-- Constraints for table `login`

--

ALTER TABLE `login`

ADD CONSTRAINT `login\_ibfk\_1` FOREIGN KEY (`username`) REFERENCES `userdetails` (`username`);

--

-- Constraints for table `reactions`

--

ALTER TABLE `reactions`

ADD CONSTRAINT `reactions\_ibfk\_1` FOREIGN KEY (`Sid`) REFERENCES `shares` (`ShareId`) ON DELETE CASCADE,

ADD CONSTRAINT `reactions\_ibfk\_2` FOREIGN KEY (`username`) REFERENCES `userdetails` (`username`);

--

-- Constraints for table `shares`

--

ALTER TABLE `shares`

ADD CONSTRAINT `shares\_ibfk\_1` FOREIGN KEY (`SType`) REFERENCES `types` (`tid`),

ADD CONSTRAINT `shares\_ibfk\_2` FOREIGN KEY (`username`) REFERENCES `userdetails` (`username`);

--

-- Constraints for table `typesconfig`

--

ALTER TABLE `typesconfig`

ADD CONSTRAINT `typesconfig\_ibfk\_1` FOREIGN KEY (`tid`) REFERENCES `types` (`tid`),

ADD CONSTRAINT `typesconfig\_ibfk\_2` FOREIGN KEY (`username`) REFERENCES `userdetails` (`username`);

-- --------------------------------------------------------

**4.2 SQL Queries**

======== SIMPLE ===========

1) "SELECT RType FROM reactions WHERE Sid="+Sid+" AND username='"+user+"'";

2) "DELETE FROM reactions WHERE Sid="+SID+" AND username='"+user+"'";

3) "UPDATE reactions SET RType='"+type+"' WHERE Sid="+SID+" AND username='"+user+"'";

4) "INSERT INTO reactions(RType,Sid,username) VALUES (?,?,?);"

5) "SELECT \* FROM shares S,userdetails U WHERE S.username=U.username AND S.ShareId="+sid;

6) "SELECT RType,fullname FROM reactions R,userdetails U WHERE R.Sid="+sid+" AND R.username=U.username";

===========================

======== NESTED ===========

7) "SELECT hashName FROM hashtag WHERE hashid IN (SELECT hashid FROM hashshares WHERE Shareid="+Sid+")";

8) SELECT MAX(cnt),RType,S.STitle FROM (SELECT COUNT(\*) as cnt,RType,Sid FROM reactions GROUP BY RType) as tmp, shares S WHERE tmp.Sid=S.ShareId GROUP BY RType;

9) SELECT ShareId FROM shares WHERE ShareId IN (SELECT Shareid FROM hashshares WHERE hashid=(SELECT hashid FROM hashtag WHERE hashName='"+srch.trim()+"'))";

10) SELECT U.fullname as Sharer,U2.fullname as Cmntr,C.cmntOn FROM comments C,userdetails U,userdetails U2 WHERE C.username = U2.username AND C.Sid = (SELECT ShareId FROM shares S WHERE S.ShareId=C.SId AND username=U.username);

11) SELECT \* FROM shares S,hashshares H WHERE H.Shareid=S.ShareId AND EXISTS (SELECT \* FROM hashtag WHERE hashName=?)

=================================

======== SET OPERATIONS ==========

12) (SELECT \* FROM shares S,userdetails U WHERE (S.username=U.username AND S.ShareId="+shareids.getInt("ShareId")+") ORDER BY SharedOn DESC)

UNION

(SELECT \* FROM shares S,userdetails U WHERE (S.username=U.username AND S.ShareId="+shareids.getInt("ShareId")+") ORDER BY SharedOn DESC);

13) (SELECT \* FROM shares S,userdetails U WHERE (S.username=U.username AND S.ShareId="+shareids.getInt("ShareId")+") ORDER BY SharedOn DESC)

INTERSECT

(SELECT \* FROM shares S,userdetails U WHERE (S.username=U.username AND S.ShareId="+shareids.getInt("ShareId")+") ORDER BY SharedOn DESC);

14) (SELECT \* FROM shares S,userdetails U WHERE (S.username=U.username AND S.ShareId="+shareids.getInt("ShareId")+") ORDER BY SharedOn DESC)

EXCEPT

(SELECT \* FROM shares S,userdetails U WHERE (S.username=U.username AND S.ShareId="+shareids.getInt("ShareId")+") ORDER BY SharedOn DESC);

=============================

====== GROUP BY ========

15) SELECT MAX(cnt),RType,S.STitle FROM (SELECT COUNT(\*) as cnt,RType,Sid FROM reactions GROUP BY RType,Sid) as tmp, shares S WHERE tmp.Sid=S.ShareId GROUP BY RType";

16) SELECT \* FROM shares GROUP BY SType;

========================

======= HAVING ========

17) SELECT \* FROM shares GROUP BY SType HAVING (username = ?);

18) SELECT \* FROM userdetails U, typesconfig T WHERE T.username=U.username GROUP BY T.username HAVING (tid = ?);

=======================

====== (NOT)/EXISTS ========

19) SELECT \* FROM shares S,hashshares H WHERE H.Shareid=S.ShareId AND EXISTS (SELECT \* FROM hashtag WHERE hashName=?);

20) SELECT \* FROM shares S WHERE NOT EXISTS (SELECT \* FROM shares WHERE SLink IS NULL);

===========================

==== AGGREGATE =======

21) SELECT COUNT(\*) INTO count FROM `comments` WHERE Sid=Shareid

22) SELECT count(\*) INTO count FROM shares WHERE username=user

23) SELECT MAX(cnt),RType,S.STitle FROM (SELECT COUNT(\*) as cnt,RType,Sid FROM reactions GROUP BY RType,Sid) as tmp, shares S WHERE tmp.Sid=S.ShareId GROUP BY RType";

24) SELECT MIN(cnt),RType,S.STitle FROM (SELECT COUNT(\*) as cnt,RType,Sid FROM reactions GROUP BY RType,Sid) as tmp, shares S WHERE tmp.Sid=S.ShareId GROUP BY RType";

======================

===== LIKE/ BETWEEN =====

25) SELECT \* FROM shares WHERE STitle LIKE "%Google%";

26) SELECT hashid FROM hashtag WHERE hashName LIKE "%Microsoft%";

27) SELECT \* FROM shares WHERE DATE(SharedOn) BETWEEN '2016-11-15' AND '2016-11-17'

=========================

===== CORRELATED =======

28) SELECT U.fullname as Sharer,U2.fullname as Cmntr,C.cmntOn FROM comments C,userdetails U,userdetails U2 WHERE C.username = U2.username AND C.Sid = (SELECT ShareId FROM shares S WHERE S.ShareId=C.SId AND username=U.username);

// used in trigger

========================

===== VIEWS ===========

usermindetails

======================

=========== TRIGGERS ===================

29) CREATE TRIGGER `onComment` AFTER INSERT ON `comments`

FOR EACH ROW BEGIN

INSERT INTO commentnotif

SELECT U.fullname as Sharer,U2.fullname as Cmntr,C.cmntOn FROM comments C,userdetails U,userdetails U2 WHERE C.username = U2.username AND C.Sid = (SELECT ShareId FROM shares S WHERE S.ShareId=C.SId AND username=U.username);

END

30) CREATE TRIGGER `postnotif` AFTER INSERT ON `shares`

FOR EACH ROW BEGIN

INSERT INTO `postnotif` VALUES (NEW.username,NEW.SharedOn);

END

=========================================

======= PROCEDURES =============

31) CREATE PROCEDURE `getIndCmntCount`(IN `Shareid` INT, OUT `count` INT)

READS SQL DATA

SELECT COUNT(\*) INTO count FROM `comments` WHERE Sid=Shareid;

32) CREATE PROCEDURE `getUserCmnts`(IN `user` VARCHAR(255), OUT `count` INT)

READS SQL DATA

SELECT COUNT(\*) INTO count FROM comments WHERE username=user;

================================

{call getIndCmntCount (?, ?)}

{call getUserShares (?, ?)}

================================