**Mingle:Your Social Network Platform**

Team:

Gauri Sarode: gss383

Joby Joy: jj2196

Office social network is a system that allows all the employee to connect and update each other with their work tasks, form teams to make projects and share pictures and location of the meetings and more relevant items.

Following are the functionalities of the system:

·    Create a profile:

Here the user creates a profile of himself. Adds his details, past work experience and educational background and co-curricular details.

Along with that the user has a security access level or job clearance level

·    Post tasks:

The user will update the task on a frequent basis, given to him by his/her boss. The task will be about the shipment he/she is handing, location of that shipment, percentage route it has already covered, and security level for that task, as in who all can view that task and who all can edit it.

·    Collaborate and form teams:

An employee can send request to other employees to form teams to do a project.

They can accept or decline the request. Once the employees belong to the same team, they can view each other’s tasks and according to the security level of both the employees and the task posted.

·    Sending response:

An employee can send a response, update or a suggestion on the tasks posted by other employees.

·    Upvoting tasks:

The employees can upvote or downvote posted tasks depending on the clearance levels both the tasks and employees have.

·    Searching for tasks/shipments:

Employees can look for tasks from the keywords and according to the security clearance they can view and get results of the resultant tasks.

Settings of the database:

·    The relation attribute response\_id records the employee name who has sent a response.

·    The relation attribute vote\_id has the employee names who has upvoted the certain task.

·    The collab\_time stores the time of the event when two employees have collaborated, and it is stored as null only if the collaboration request for a project is unanswered.

Assumptions:

* Every employee is assigned a level between A to D where A refers to Junior associates, B refers to senior associates, C refers to managers, D refers to CEO, President, Trustee. By default a new employee is assigned the level A following which the HR assigns the corresponding level based on their skills and experience.
* Access id are allowed at the profile and post level.
* Access id’s can be S: Self, T: Team, F: Friends, FOF : Friend and Friends of Friends, P: Public.

Relational Schema:

* employee(**ssn,**email\_id)
* signin(ssn,password)
* registered\_employee(**profile\_name**,ssn,first\_name,last\_name,address,designation,skills,interests,manager,level,access\_id,email\_id,profile\_pic,gender)
* post(**post\_id**,post\_date,post\_title,post\_desc,access\_id)
* wall(post\_id,profile\_name)
* relationship(sender\_name,receiver\_name,relation\_type,request\_time,request\_status)
* multimedia\_content(**multimedia\_id**,post\_id,multimedia\_name,multimedia\_type, multimedia\_content,access\_id)
* comment(**comment\_id**,post\_id,desc,comment\_date,commentor\_name,access\_id)
* location(**loc\_id,**post\_id,name,description,latitude,longitude,access\_id)
* likes(**like\_id**,post\_id,comment\_id,multimedia\_id,loc\_id,viewer\_name,like\_date)

Foreign Keys:

* ssn in signin table is a foreign key referencing the ssn in the employee table.
* ssn in the registered\_employee table is a foreign key referencing the ssn in employee table.
* post\_id,profile\_name  in wall table references the  post\_id in post and profile\_name in the registered\_employee table respectively.
* In multimedia\_connent table post\_id is the foreign key referencing table post.
* In comment table post\_id is the foreign key referencing table post and commentor\_name is the foreign key referencing table registered\_employee.

* In like table likes viewer\_name is the foreign key referencing table registered\_employee and post\_id is the foreign key referencing table post and multimedia\_id is the foreign key referencing table multimedia\_content and loc\_id is the foreign key referencing table location.
* In relationship table sender\_name references table registered\_employee.

**DDL Statement:**

create database Mingle;

use Mingle;

drop table if exists employee;

#Create employee table

create table employee(

ssn int primary key, email\_id varchar(20) not null

);

#Create sign in table

drop table if exists signin;

create table signin

(

ssn int unique,

password varchar(50) not null,

foreign key (ssn) References employee(ssn)

on delete cascade

);

#Create registered\_employee  table

drop table if exists registered\_employee;

create table registered\_employee

(

profile\_name varchar(30) primary key,

ssn int unique,

first\_name varchar(20) not null ,

last\_name varchar(20) not null,

address varchar (255),

designation varchar(20),

skills varchar(255),

interests varchar(255),

manager varchar(20),

level enum('A','B','C','D') default 'A',

email\_id varchar(50),

profile\_pic blob,

gender enum('Male','Female'),

access\_id enum('P','T','F','S','FOF') default 'S',

foreign key (ssn) References employee(ssn)

on delete cascade

);

#Create post

drop table if exists post;

create table post

(

post\_id varchar(30),

post\_time timestamp not null,

post\_title varchar(150),

post\_desc varchar(255),

access\_id varchar(5) default 'S',

primary key (post\_id)

);

#Create wall(post\_id,profile\_name)

drop table if exists wall;

create table wall

(

post\_id varchar(20),

profile\_name varchar(30),

foreign key (profile\_name) references registered\_employee(profile\_name)

on delete cascade

on update cascade,

foreign key (post\_id) references post(post\_id)

on delete cascade

on update cascade

);

#Create relationship table

drop table if exists relationship;

create table relationship

(

sender\_name varchar(30),

receiver\_name varchar(30),

relation\_type varchar(20),

request\_time timestamp,

request\_status enum('Accepted','Declined'),

foreign key (sender\_name) references registered\_employee(profile\_name)

on delete cascade

on update cascade,

foreign key (receiver\_name) references registered\_employee(profile\_name)

on delete cascade

on update cascade,

primary key(sender\_name,receiver\_name)

);

#Create multimedia table

drop table if exists multimedia\_content;

create table multimedia\_content

(

multimedia\_id varchar(10) primary key,

post\_id varchar(20),

multimedia\_name varchar(20),

Multimedia\_type varchar(20),

multimedia\_content blob,

access\_id varchar(5) default 'S',

foreign key (post\_id) references post(post\_id)

on delete cascade

on update cascade

);

#Create table Comment

drop table if exists comment;

create table comment

(

comment\_id varchar(20) primary key,

post\_id varchar(20),

comment\_desc varchar(255),

comment\_date datetime,

commentor\_name varchar(30),

access\_id varchar(5) default 'S',

foreign key (commentor\_name) references registered\_employee(profile\_name)

on delete cascade

on update cascade,

foreign key (post\_id) references post(post\_id)

on delete cascade

on update cascade

);

#Create location table

drop table if exists location;

create table location

(

loc\_id varchar(20) primary key,

post\_id varchar(20),

name varchar(100),

description varchar(255),

latitude varchar(30),

longitude varchar(30),

access\_id varchar(5) default 'S',

foreign key (post\_id) references post(post\_id)

on delete cascade

on update cascade

);

#Create like table

drop table if exists likes;

create table likes

(

like\_id varchar(20) primary key,

post\_id varchar(30),

multimedia\_id varchar(10),

comment\_id varchar(20),

loc\_id varchar(20),

viewer\_name varchar(20),

like\_time datetime,

foreign key (post\_id) references post(post\_id) on delete cascade on update cascade,

foreign key (multimedia\_id) references multimedia\_content(multimedia\_id) on delete cascade on update cascade,

foreign key (comment\_id) references comment(comment\_id) on delete cascade on update cascade,

foreign key (loc\_id) references location(loc\_id) on delete cascade on update cascade,

foreign key (viewer\_name) references registered\_employee(profile\_name) on delete cascade on update cascade

);