Share, Contribute, and Comment System

DRAFT REPORT

Submit by:

Olivier Racette – 40017231

Xavier Vani-Charron – 27055838

Rami El-Kazma – 40035141

Keven Presseau-St-Laurent – 40000501

Louis-Simon Carle – 26677266

Submit to:

Dr. Bipin C. Desai

Concordia University

Gina Cody School of Engineering and Computer Science

Submission Date:

December 2nd, 2019

TABLE OF CONTENTS

1 Project Description

1.1 Introduction

Our application is a share, contribute, and comment (SCC) system, that connects users in a way that resembles a social media platform. The application gives users the ability to attend events and join groups including other participants attending the event, where they can share content such as text, images, videos, etc. It also allows members to comment on the content and to communicate directly and privately with one another, if they so wish. The SCC is an interactive platform that users can view as a virtual society.

1.2 Scope

The application is divided into three aspects: a web-based user interface implemented with HTML, CSS, and JavaScript that allows the user to log in and perform all the supported actions, a server-side implemented with PHP encoding the functionality of the website, and a relational database management system, MySQL, to manage the application's database that includes all the stored information of the user, event, group, etc. The system handles queries initiated by user actions at the client-side and processed at the server-side resulting in manipulation or retrieval of the data from the database.

1.3 Objective

The objective is to design the SCC system as described (see Section 1.1) adhering to the following specific requirements. The SCC is managed by a system administrator that is also responsible functionality of events. Event managers register and oversee events, and they can manage multiple events. They also add or remove participants from the event, who are users that can have different roles in the SCC. Participants can create groups within specific events, becoming a group manager, which are open to all of its participants through invitations only. However, event participants can view and request to join groups associated with the event. Event participants and group members can share content (text, images, or videos) with other participants and comment on them. Events are active for a certain period, at which point they are archived keeping all the basic details.

2 Assumptions

To begin, a user needs a HTML-5 compliant web browser such as Google Chrome, Mozilla Firefox, Apple Safari, etc. Then, to be able to use this website the user needs the proper URL. This project is currently only being hosted on (https://hrc353.encs.concordia.ca/). Once the user accesses the website for the first time, he will need to sign up by completing the registration form and creating a username and password. Once the registration is completed, the user will be able to gain regular access to the website. However, to gain complete access to the website, an administrator needs the username and password for an account that has been granted admin privileges that is currently stored in the user database hosted by the server.

3 Limitations

Since most of the code for this project is executed on the server, the only possible limitations are client-side incompatibility. In other words, any browser that is currently not up to date is not guaranteed to be able to run this application properly, because this project uses HTML-5, CSS and a bit of JavaScript. Another possible limitation would be a user with a compliant browser but without access to a physical or digital keyboard, because this project requires a keyboard simply to be able to log into a user's own account.

4 Applications Supported

4.1 Text Version

The current iteration of this project is currently aimed at desktop internet browsers. As such, it has been tested on the most used browsers: Google Chrome, Mozilla Firefox, Apple Safari and Microsoft Edge (https://gs.statcounter.com/browser-market-share/desktop/worldwide). However, this project uses the proper viewports and CSS tables to be compatible on mobile devices, but testing has not been extensively done on such platforms.

4.2 List Version

PC:

Chrome (tested on version 78.0.3904.108)

Firefox (tested on version 70)

Safari (tested on version 13.0.2)

Edge (tested on version 44.18362.449.0)

5 Architectural Design

5.1 Architectural Style

The SCC system is designed as a database with a Relational DBMS that acts on the data based on the interactions of users with the system. We chose the Data-Centered Architecture (see Figure 1) for the application because it entails data that is centralized and accessed frequently by the clients. The central data is shared, and clients interact with each other through the database; meaning that each client performs actions that affect that state of the database which can then be accessed by other clients. Therefore, clients are relatively independent of each other, only interacting through the data store.

The data-centered architecture has two components:

- Central Data Structure: the database, which stores data in a set of relations and their attributes interacting with each other through relationships.
- Data Accessors: the set users of the system with different roles who access data to retrieve or modify it, changing the state of the central data structure.

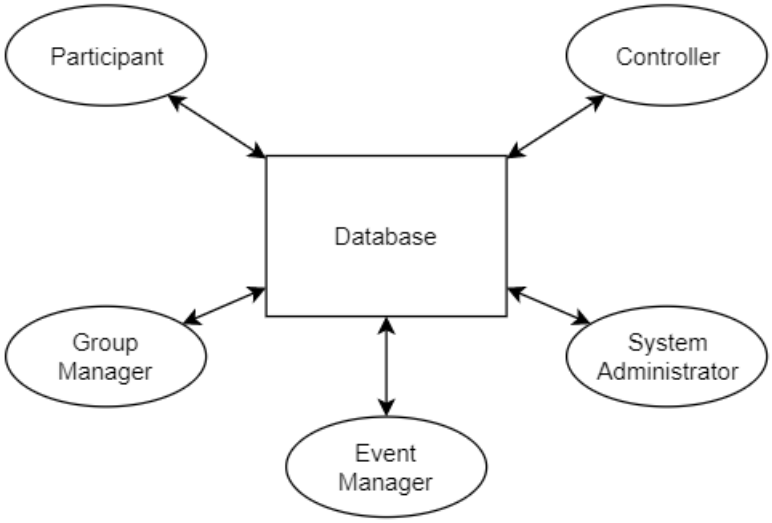


Figure 1. Data-Centered Architecture for the SCC System

To explain in more detail, we used a Repository Architecture Style, where the clients actively change the data in the database and then check for changes. The data structure is passive, and the data accessors control the changes in the data.

5.2 Architectural Design Pattern

We used the MVC (Model, View, Controller) architectural design pattern to implement the user interface of the application and its interaction with the system.

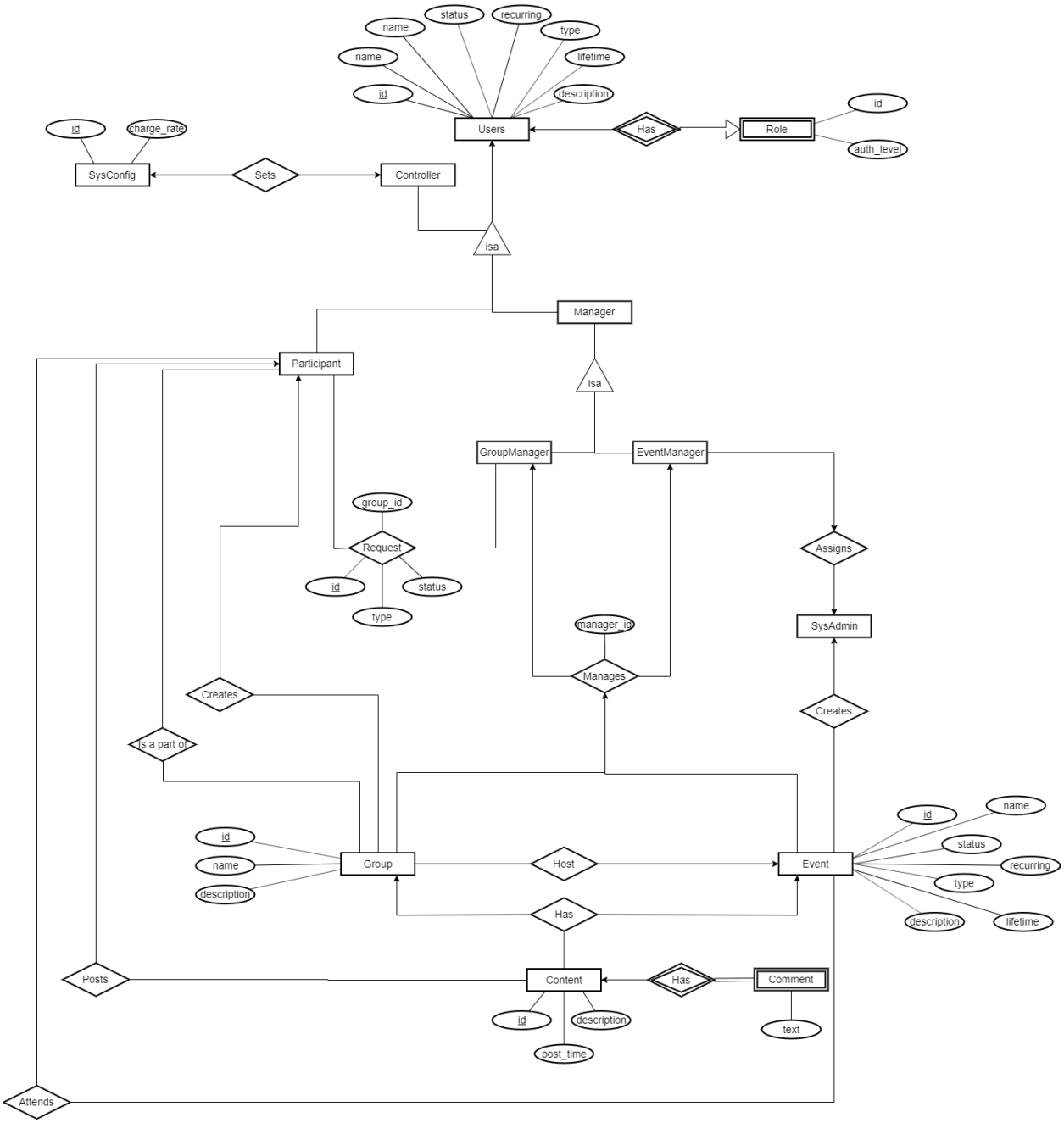
The data models are the classes that resemble the relations in our database. When a controller makes a change in an event time for example, the 'lifetime' data member of the event class will be updated, then querying the database to change the 'lifetime' attribute of the Event relation for that specific event.

The controller makes the changes required based on input from the user. For example, when an event manager changes the lifetime of an event, the event controller will update the Event class.

The views are the different pages that a user view when using the system. For example, when an event's time is updated, the user's homepage will show the new time of this event.

6 E-R Diagram

We included a clearer picture (see E-R Diagram.png) or for an editable version,



7 Relational Database Design

7.1 Definition of Relations, Attributes, and Foreign Keys

The relations and attributes that we have defined are the following:

- Users – this entity represents any of the various human users of the system; the system administrator, a controller, event managers, group managers, and participants.
    - ID
    - Role_ID – foreign key referencing ID of the Role relation.
    - Username, User_Password, Email, First_Name, Last_Name, Adr_Number, Apt_Number, Street, City, Dob
- Role – this entity represents the role that a User has.
    - ID
    - User_Role, Auth_Lvl
- Organizations – this entity represents the organization hosting an event.
    - ID
    - User_ID – foreign key referencing ID of the User relation.
    - Name, Type
- Debit_Details – this entity represents the bank information of a User.
    - ID
    - User_ID – foreign key referencing ID of the User relation.
    - Bank_Num, Account_Code
- Events – this entity represents an event that a User manages and a User can attend.
    - ID
    - Manager_ID – foreign key referencing ID of the User relation.
    - Event_Name, Event_Status, Fee, Event_Description, Lifetime, Recurring.
- Event_Participants – this entity represents a User that attends an Event.
    - User_ID – foreign key referencing ID of the User relation.
    - Event_ID – foreign key referencing ID of the Event relation.
- Groups – this entity represents a group that a User manages and a User can join.
    - ID
    - Event_ID – foreign key referencing ID of the Event relation.
    - Manager_ID – foreign key referencing ID of the User relation.
    - Name, Description
- Group_Participants – this entity represents a User that joins a Group.
    - User_ID – foreign key referencing ID of the User relation.
    - Group_ID – foreign key referencing ID of the Group relation.

- Content – this entity represents any of the content types that a User can post.
    - ID
    - User_ID – foreign key referencing ID of the User relation.
    - Type, Content, Post_Time
- Event_Content – this entity represents a Content that a User posts to an Event.
    - ID
    - User_ID – foreign key referencing ID of the User relation.
    - Event_ID – foreign key referencing ID of the Event relation.
- Group_Content – this entity represents a Content that a User posts to a Group.
    - Content_ID – foreign key referencing ID of the Content relation.
    - User_ID – foreign key referencing ID of the User relation.
    - Group_ID – foreign key referencing ID of the Group relation.
- Content_Comment – this entity represents a comment that a User posts on a Content.
    - ID
    - Content_ID – foreign key referencing ID of the Content relation.
    - User_ID – foreign key referencing ID of the User relation.
    - Text
- Requests – this entity represents a request to join a Group sent by one User to another User.
    - ID
    - Source_ID – foreign key referencing ID of the User relation.
    - Dest_ID – foreign key referencing ID of the User relation.
    - Group_ID – foreign key referencing ID of the Group relation.
    - Type, Status
- Messages – this entity represents a direct message that one User sends another User.
    - ID
    - Source_ID – foreign key referencing ID of the User relation.
    - Dest_ID – foreign key referencing ID of the User relation.
    - Text, Sent_Time
- System_Config – this entity
    - ID
    - Charge_Rate

7.2 Primary Keys

We implemented our database such that every table relation except Event_Participants, Group_Participants, Event_Content, and Group_Content have an ID attribute that uniquely defines that relation. All other attributes are fully functionally dependent on the attributes, ID, which are the only primary keys of their respective tables (ex. primary key of Users is ID, all other attributes are dependent on ID).

The primary keys of the above–mentioned exceptions are all the attributes of the relations (ex. primary key of Event_Participants is User_ID and Event_ID).

7.3 Relationships

The relationships between different entities are defined generally through foreign keys (see Section 7.1) and in more detail, specifically the types (ex. one–to–one), in the ER–Diagram (see Section 6).

8 3NF Solution

8.1 Relations

Users

ID	Role_ID	Details

- ID is the primary key of the table.[1]
- Details and Role_ID depend on ID.[2]

Role

ID	User_Role	Auth_Lvl

- ID is the primary key of the table.[1]
- User_Role and Auth_Lvl depend on ID.

Organizations

ID	User_ID	Name	Type

- ID is the primary key of the table.[1]
- Owner_ID is a foreign key referring to ID in the Users table and depends on ID.
- Name and Type depend on ID.

Debit_Details

ID	User_ID	Bank_Num	Account_Code

- ID is the primary key of the table.[1]
- User_ID, Bank_Num, and Account_Code depend on ID.

[1] ID attribute corresponds to the identity of the entity.
[2] Details corresponds to the non-key attributes that are only dependent on the ID (username, user_password, email, first_name, last_name, adr_number, apt_number, street, city, dob) for Users and (event_name, event_status, fee, event_description, lifetime, recurring) for Events (not included in the interest of space and clarity).

Events

ID	Manager_ID	Details

- ID is the primary key of the table.[1]
- Manager_ID and Details depends on ID.[2]

Event_Participants

User_ID	Event_ID

- User_ID, Event_ID is the primary key of the table.

Groups

ID	Event_ID	Manager_ID	Name	Description

- ID is the primary key of the table.[1]
- Event_ID, Manager_ID, Name, and Description depend on ID.

Group_Participants

User_ID	Group_ID

- User_ID, Group_ID is the primary key of the table.

Content

ID	User_ID	Type	Content	Post_Time

- ID is the primary key of the table.[1]
- User_ID, Type, Content, and Post_Time depend on ID.

Event_Content

Content_ID	User_ID	Event_ID

- Content_ID, User_ID, Event_ID is the primary key of the table.

Group_Content

Content_ID	User_ID	Group_ID

- Content_ID, User_ID Group_ID is the primary key of the table.

Content_Comment

ID	Content_ID	User_ID	Text

- ID is the primary key of the table.[1]
- Content_ID, User_ID, and Text depend on ID.

Requests

ID	Source_ID	Dest_ID	Group_ID	Type	Status

- ID is the primary key of the table.[1]
- Source_ID, Dest_ID and Group_ID, Type, and Status depend on ID.

Messages

ID	Source_ID	Dest_ID	Text	Sent_Time

- ID is the primary key of the table.[1]
- Source_ID, Dest_ID, Text, and Sent_Time depend on ID.

System_Config

ID	Charge_Rate

- ID is the primary key of the table.[1]
- Charge_Rate depends on ID.

8.2 Normalization

1st Normal Form

- Each relation has a primary key on which all other attributes depend.
- All attribute values are atomic ie. every tuple consists of a single value for each attribute in the relation.
- All attribute values are of the same type.

2nd Normal Form

- The relations in this database are in first normal form.
- There are no partial dependencies in any relation. All non–prime attributes, which in this case are all attributes that are not the primary key (ID) in each relation are fully functionally dependent on the primary key, ID.

3rd Normal Form

- The relations in this database are in first normal form and second normal form.
- There are no transitive dependencies in any relation. There are not non–prime attributes that are functionally dependent on another non–prime, which in this case are all attributes that are not the primary key, ID. All attributes are only functionally dependent on the primary key, and not on another non–prime attribute.

9 Member Responsibility

Xavier Vani-Charron worked on back-end and front-end development.

Olivier Racette worked on SQL scripts and back-end development.

Rami El-Kazma worked on SQL scripts and the report.

Louis-Simon Carle worked on SQL scripts and back-end development, and the E-R Diagram.

Keven Presseau-St-Laurent worked on front-end development and the report.

10 Gantt Chart

11 Detailed Analysis of the Coding the Website

We used the MVC architectural design patter as the backbone of our website (see Section 5.2).

11.1 The Model

We implemented the models as a database relation representation with data members resembling attributes. Any changes that are to be made on the database, such as updating a first name for example, will be made on that specific user object's first name data member, which will then query the database to make the proper change. It is not a one-to-one relationship between models and database relations however, because it would prove to be redundant. For example, the link between an event and its content is done through the event_content relation in the database but we only implemented Content is a data model. This is because dividing the data model into event_content and group_content would be less efficient then simply having the 'Content' data model query the database for event and group content.

11.2 The Controller

We implemented controllers for each 'action' that a user can do that will make the necessary changes to fulfill the user's task. For example, the login and logout controllers will start and stop the session respectively when a user clicks on the corresponding button. The event controller will display all the events that a user is participating in and managing when they access the events page through the tab on the navigation bar.

11.3 The View

We implemented all the pages that a user can access through the website as the views. For a more detailed analysis of the interface itself (see Section 12) or for an overview (see Section 18). The view is the display of the website, and so the implementation of the Groups page for example is a list of all the groups that a user has joined and ones that they are managing. The view of a selected group would be a list of participants in that group with the ability to message them and the content with comments that other participants have posted.

12 The Interface Design Rational

The interface is designed in a way to make it as user friendly as possible. We used the Bulma CSS framework as a template for our website. We used the navigation bar to make it as easy possible for a user to navigate between the different pages that they have access to. The main section of the pages only includes sections that are relevant to the user at the current view (ex. Events page only includes events managed and events participating in) to keep it as compact and clear as possible. This however does not take away from functionality as the user will have ease of access to all relevant issues when using the website. The path to a certain page will be clear. For example, to access post a comment on the content of a certain event, a user will navigate to the event which will display the content posts allowing the user to easily add a comment.

13 Queries

```
--CREATE A USER

INSERT INTO users (role_id, username, user_password, email, first_name, last_name,

                adr_number, street, city, dob)

VALUES (roleID, userName, userPass, email, firstName, lastName,

        adr_number, street, city, bday);


--DELETE A USER

DELETE FROM users

WHERE id = id;


--EDIT A USER

--User can edit any column except IDs and then we set the updates in php using
setters and update all variables in mysql (whatever didn't change will just stay the
same)

UPDATE users

SET username = userName, user_password = userPass, email= email, first_name =
firstName, last_name = lastName, adr_number = adr_number, street = street, city =
city, dob = bday

WHERE id = id;


--DISPLAY A USER

--Display by ID, username, or FULL name

SELECT * FROM users

WHERE (id = id OR username = username

              OR (first_name = first_name AND last_name = last_name));


--CREATE A GROUP

INSERT INTO groups (event_id, manager_id, group_name, group_description)

VALUES (event_id, manager_id, name, description);


--DELETE A GROUP
```

```
DELETE FROM groups

WHERE (id = id);


--EDIT A GROUP

--User can edit any column except IDs and then we set the updates in php using
setters and update all variables in mysql (whatever didn't change will just stay the
same)

UPDATE groups

SET group_name = name, group_description = description

WHERE id = id;


--DISPLAY A GROUP

--Display by ID, name, or any string that matches part of the descritpion

SELECT * FROM groups

WHERE (id = id OR name = name OR description LIKE '%description%');


--CREATE A LIST OF MEMBERS IN A GROUP


--DELETE A LIST OF MEMBERS IN A GROUP


--EDIT A LIST OF MEMBERS IN A GROUP


--DISPLAY A LIST OF MEMBERS IN A GROUP


--REQUEST TO JOIN A GROUP IN THE EVENT

--Assumption is this is requested by a user object (ie. id is the user's ID)

--mgrOfGroup - get ID of the group manager

INSERT INTO requests (source_id, dest_id, group_id, request_type, request_status)

VALUES (id, mgrOfGroup, groupID, type, false);


--WITHDRAW FROM A GROUP

--Assumption is this is done by a user object (ie. id is the user's ID)
```

```
DELETE FROM group_participants
WHERE id = id;

--POST TEXTS, IMAGES, OR VIDEOS
--Assumption is this is done by a user object (ie. id is the user's ID)

--First insert into content table
INSERT INTO content (user_id, type, content, post_time)
VALUES (id, type, content, post_time);

--Insert into event_content
INSERT INTO event_content (content, user_id, event_id)
VALUES (content, id, eventID);
--Insert into group_content
INSERT INTO group_content (content, user_id, group_id)
VALUES (content, id, groupID);

--VIEW POSTS BY OTHER MEMBERS
--Assumption is this is done by a user object (ie. id is the user's ID)
--Group
SELECT content, post_time
FROM content c
INNER JOIN group_content gc
ON c.id = gc.content_id
INNER JOIN group_participants gp
ON gc.group_id = gp.group_id
WHERE gp.user_id = id;

--Event
SELECT content, post_time
FROM content c
INNER JOIN event_content ec
```

```
ON c.id = ec.content_id
INNER JOIN event_participants ep
ON ec.group_id = ep.group_id
WHERE ep.user_id = id;

--COMMENT ON POSTS
--Assumption is this is done by a user object (ie. id is the user's ID)

INSERT INTO content_comments (content_id, user_id, text)
VALUES (group_contentID, id, text);

--VIEW COMMENTS ON POSTS
--Assumption is this is done by a user object (ie. id is the user's ID)



SELECT text
FROM content_comments cc
INNER JOIN group_content gc
ON cc.content_id = gc.content_id
WHERE gc.content_id = content_id;

--SEND PRIVATE MESSAGE
--Assumption is this is done by a user object (ie. id is the user's ID)

--List of group members
SELECT user_id
FROM group_participants gc1
WHERE gc1.group_id = (SELECT group_id
                    FROM group_participantts gc2
                    WHERE gc2.id = id)

--SEND THE MESSAGE
```

```
INSERT INTO messages (source_id, dest_id, text, sent_time)
VALUES (id, destID, msg, time);
```

14 User Manual

Login: Enter username and password

Sign Up: Enter all required details

Forgot Password: Enter username and a link to reset password will be sent by email.

Update information: My Account > Update the info

Create a Group: Groups > Create a Group > Enter required details.

Request to Join a Group: Events > View > Choose a group from the list of groups

Add Content: Events / Groups > Select the event/group > Post the content

Comment on Content: Events / Groups > Select the event/group > Content > Comment on the post.

Message: Groups > Select a group > Participants > Message > Write the message and send.

Withdraw from Group: Groups > Leave Group (group that you wish to leave)

15 General Information

This application is an online communication board used to organize events between users. It allows users to create and manage events, create and manage groups inside these events and discuss with other users that are part of the same group or event. It also allows users to exchange images, videos and text inside those discussions or privately with each other.

16 Authorized Use Permissions

17 System Summary

Users:

- Admins: Maintains this application, creates events and assigns event managers.
- Controllers: Sets pricing for events and resource management
- Participant: Participates in events and joins groups.

Events: Created by admins and managed by participants that are event managers

Groups: Created inside an event to offer a private discussion board for a part of the participants in an event.

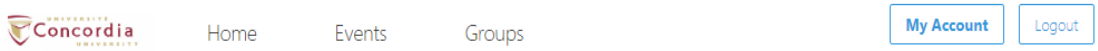
Inbox: Allows users to send messages with or without attachments to other users privately and to receive messages sent to all participants in an event.

18 Overview of the User Interface

In the sections the follow, we will have a more detailed look at the complete user interface with the different sections based on the roles of the users.

Once successfully logged in, a user's pages are divided into two sections:

- The Navigation Bar

  The Events and Groups tab lead to the events and groups pages of the user.

  The My Account tab leads to the user's account details where they can make changes to their information.

  The Home tab redirects the user back to their homepage.

  The Logout tab logs the user out of the system.

- The Main Page

  Depending on the page, it will contain the necessary information relating to that particular page. For example, the 'My Account' page will include a user's account details, while the 'Create a Group' page will prompt the user to enter the necessary information to create a group.

19 Home Page

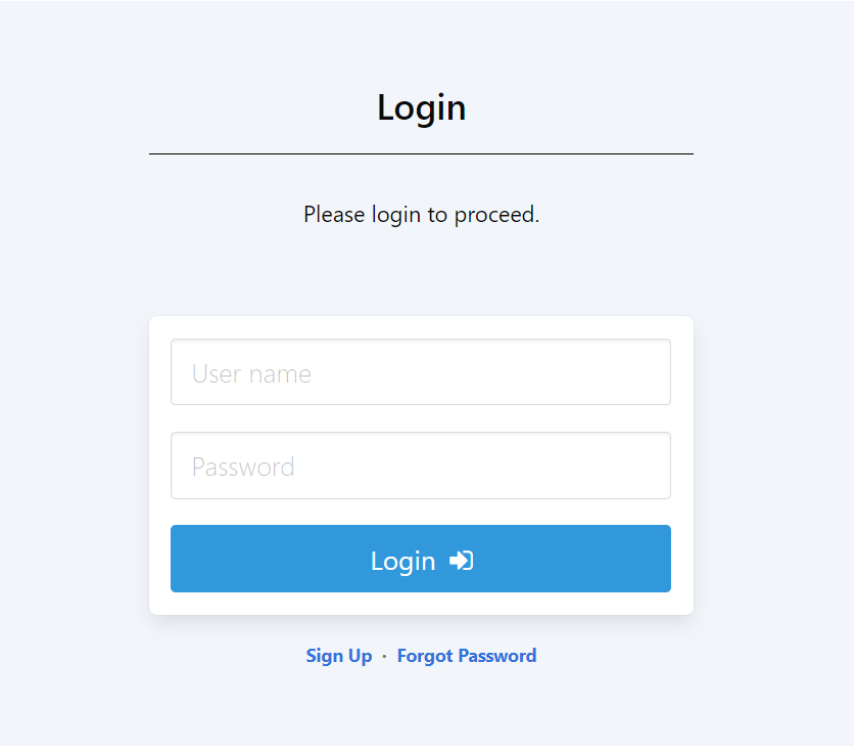


Figure 2. Login Page

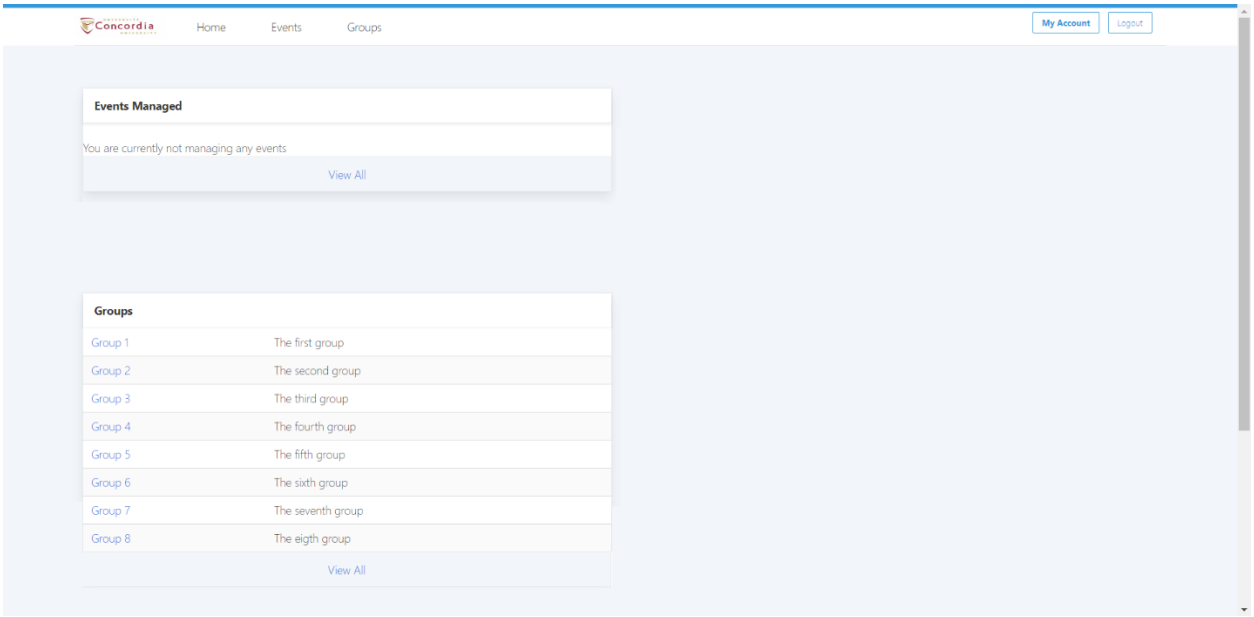


Figure 2. Home Page

20 Visitors

The application requires that a user is always logged in in order to view the content as it is a "private virtual society". Therefore, a person that accesses the website through the URL will be prompted to either log-in using a unique username–password combination or register to the site to create these credentials. In order to sign-up the user will be asked to input certain details to use the system.

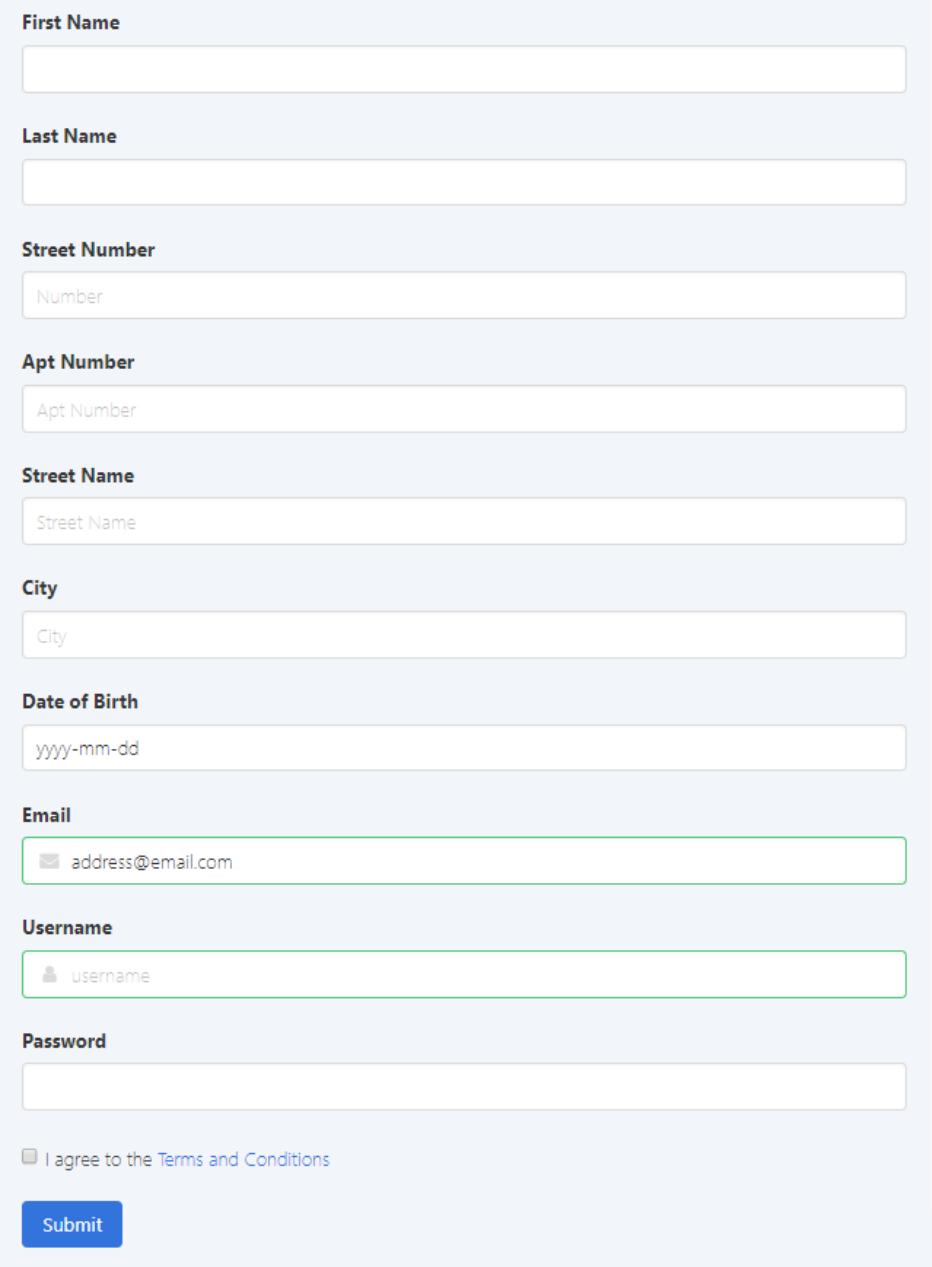
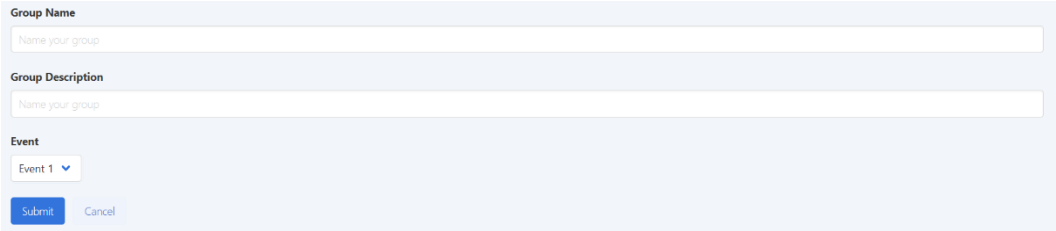


Figure 3. Registration Page

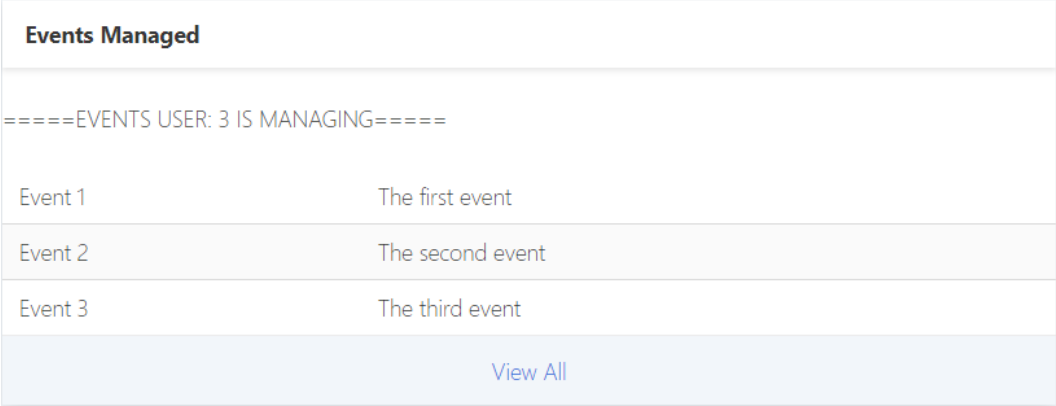
21 Members

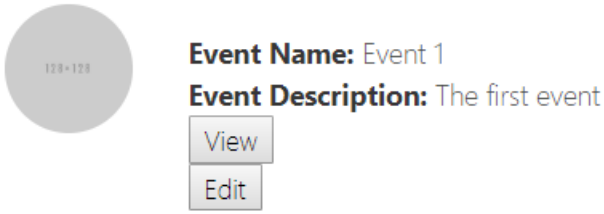
Members who access the site will be directed to the homepage where they can enter their login information (username and password). Apart from the administrator, there are three types of members:

- Participant: The simplest type of member who is only part of events or groups.
    - Participants can see the events that they are participating in and the groups they are a part of on the home page (see Figure 1).
    - They can access the groups or events that they are participating in through the proper tab of the navigation bar at the top of the page.
    - In addition to a list of groups, the Groups page also allows the participants to create a group, where they can input a name and description, followed by the selection of the event to which the group belongs.



- Event Manager:
    - An event manager's home page will have the events that they are currently managing in the 'Events Managed' section where they can access the event and manage it based on the requirements (see Section 1.3).





- Group Manager: a user who has created a group within an event.
    - The group manager will be able to see their groups and access them to manage based on the requirements (see Section 1.3).

22 Administrator

A system administrator is a user that has privileges over all other users. They are the ones that maintain and configure the SCC.

- The system administrator's event page will not show that they are managing all events, but they will be able add, delete, or update the users in an event. They are also able to create events and assign another user as an event manager.
- The system administrator will also not actively manage all groups but they can add, delete, or edit the member of a group. They can also create a group and act as a group manager.
- The system administrator can also post content in any groups or events.

23 Git Log

See git.log file in the project submission root folder.

24 Database

We will include figures of our database accessed on the server.

```
mysql> show tables;
+------------------------+
| Tables_in_hrc353_2     |
+------------------------+
| debit_details          |
| event_content          |
| event_content_comments |
| event_participants     |
| events                 |
| group_content          |
| group_content_comments |
| group_participants     |
| messages               |
| organizations          |
| requests               |
| roles                  |
| system_config          |
| user_groups            |
| users                  |
+------------------------+
15 rows in set (0.00 sec)
```

```
mysql> desc debit_details;
+--------------+---------+------+-----+---------+----------------+
| Field        | Type    | Null | Key | Default | Extra          |
+--------------+---------+------+-----+---------+----------------+
| id           | int(11) | NO   | PRI | NULL    | auto_increment |
| user_id      | int(11) | NO   | MUL | NULL    |                |
| bank_num     | int(11) | NO   |     | NULL    |                |
| account_code | int(11) | NO   |     | NULL    |                |
+--------------+---------+------+-----+---------+----------------+
4 rows in set (0.01 sec)

mysql> desc event_content;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| event_id  | int(11)      | NO   | MUL | NULL    |                |
| content   | varchar(250) | NO   |     | NULL    |                |
| post_time | datetime     | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)

mysql> desc event_participants;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| user_id  | int(11) | NO   | MUL | NULL    |       |
| event_id | int(11) | NO   | MUL | NULL    |       |
+----------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc events;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| id                | int(11)      | NO   | PRI | NULL    | auto_increment |
| manager_id        | int(11)      | NO   | MUL | NULL    |                |
| event_name        | varchar(50)  | NO   |     | NULL    |                |
| event_status      | tinyint(1)   | YES  |     | NULL    |                |
| fee               | double       | YES  |     | NULL    |                |
| event_description | varchar(250) | YES  |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

```
mysql> desc group_content;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| group_id  | int(11)      | NO   | MUL | NULL    |                |
| content   | varchar(250) | NO   |     | NULL    |                |
| post_time | datetime     | NO   |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> desc group_participants;
+----------+---------+------+-----+---------+-------+
| Field    | Type    | Null | Key | Default | Extra |
+----------+---------+------+-----+---------+-------+
| user_id  | int(11) | NO   | MUL | NULL    |       |
| group_id | int(11) | NO   | MUL | NULL    |       |
+----------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> desc messages;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| id           | int(11)      | NO   | PRI | NULL    | auto_increment |
| source_id    | int(11)      | NO   | MUL | NULL    |                |
| dest_id      | int(11)      | NO   | MUL | NULL    |                |
| message_text | varchar(250) | NO   |     | NULL    |                |
| sent_time    | datetime     | NO   |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> desc organizations;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| user_id  | int(11)     | NO   | MUL | NULL    |                |
| org_name | varchar(50) | NO   |     | NULL    |                |
| org_type | varchar(50) | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> desc requests;
+----------------+------------+------+-----+---------+----------------+
| Field          | Type       | Null | Key | Default | Extra          |
+----------------+------------+------+-----+---------+----------------+
| id             | int(11)    | NO   | PRI | NULL    | auto_increment |
| source_id      | int(11)    | NO   | MUL | NULL    |                |
| dest_id        | int(11)    | NO   | MUL | NULL    |                |
| group_id       | int(11)    | NO   | MUL | NULL    |                |
| request_type   | varchar(7) | NO   |     | NULL    |                |
| request_status | tinyint(1) | NO   |     | NULL    |                |
+----------------+------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

```
mysql> desc roles;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| id        | int(11)     | NO   | PRI | NULL    | auto_increment |
| user_role | varchar(50) | NO   |     | NULL    |                |
| auth_lvl  | int(11)     | NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> desc system_config;
+-------------+---------+------+-----+---------+----------------+
| Field       | Type    | Null | Key | Default | Extra          |
+-------------+---------+------+-----+---------+----------------+
| id          | int(11) | NO   | PRI | NULL    | auto_increment |
| charge_rate | double  | YES  |     | NULL    |                |
+-------------+---------+------+-----+---------+----------------+
2 rows in set (0.01 sec)

mysql> desc user_groups;
+-------------------+--------------+------+-----+---------+----------------+
| Field             | Type         | Null | Key | Default | Extra          |
+-------------------+--------------+------+-----+---------+----------------+
| id                | int(11)      | NO   | PRI | NULL    | auto_increment |
| event_id          | int(11)      | NO   | MUL | NULL    |                |
| manager_id        | int(11)      | NO   |     | NULL    |                |
| group_name        | varchar(50)  | YES  |     | NULL    |                |
| group_description | varchar(250) | YES  |     | NULL    |                |
+-------------------+--------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> desc users;
+---------------+-------------+------+-----+---------+----------------+
| Field         | Type        | Null | Key | Default | Extra          |
+---------------+-------------+------+-----+---------+----------------+
| id            | int(11)     | NO   | PRI | NULL    | auto_increment |
| role_id       | int(11)     | NO   | MUL | NULL    |                |
| username      | varchar(50) | NO   | UNI | NULL    |                |
| user_password | varchar(40) | NO   |     | NULL    |                |
| email         | varchar(50) | NO   |     | NULL    |                |
| first_name    | varchar(50) | NO   |     | NULL    |                |
| last_name     | varchar(50) | NO   |     | NULL    |                |
| adr_number    | int(11)     | NO   |     | NULL    |                |
| apt_number    | int(11)     | YES  |     | NULL    |                |
| street        | varchar(50) | NO   |     | NULL    |                |
| city          | varchar(50) | NO   |     | NULL    |                |
| dob           | date        | NO   |     | NULL    |                |
+---------------+-------------+------+-----+---------+----------------+
12 rows in set (0.00 sec)
```

25 Script to Create Database and Populate It

25.1 Creating the Database

```
CREATE DATABASE IF NOT EXISTS hrc353_2 CHARACTER SET utf8 COLLATE utf8_unicode_ci;
USE hrc353_2;
SET default_storage_engine=INNODB;

-- The password field will be SHA1 encrypted

CREATE TABLE system_config (
    id INT PRIMARY KEY AUTO_INCREMENT,
    charge_rate DOUBLE
);

CREATE TABLE roles (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_role VARCHAR(50) NOT NULL,
    auth_lvl int NOT NULL
);

CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    role_id INT NOT NULL,
    username VARCHAR(50) NOT NULL,
    user_password VARCHAR(40) NOT NULL,
    email VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    adr_number INT NOT NULL,
    apt_number INT,
    street VARCHAR(50) NOT NULL,
    city VARCHAR(50) NOT NULL,
```

```sql
    dob DATE NOT NULL,
    FOREIGN KEY (role_id) REFERENCES roles(id) ON DELETE CASCADE,
    UNIQUE (username)
);

CREATE TABLE organizations (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    org_name VARCHAR(50) NOT NULL,
    org_type VARCHAR(50) NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE debit_details (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    bank_num INT NOT NULL,
    account_code INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE events (
    id INT PRIMARY KEY AUTO_INCREMENT,
    manager_id INT NOT NULL,
    event_name VARCHAR(50) NOT NULL,
    event_status BOOLEAN,
    fee DOUBLE,
    event_description VARCHAR(250),
    lifetime DATETIME NOT NULL,
    recurring BOOLEAN NOT NULL DEFAULT 0,
    FOREIGN KEY (manager_id) REFERENCES users(id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE event_participants (
    user_id INT NOT NULL,
    event_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (event_id) REFERENCES events(id) ON DELETE CASCADE
);

CREATE TABLE groups (
    id INT PRIMARY KEY AUTO_INCREMENT,
    event_id INT NOT NULL,
    manager_id INT NOT NULL,
    group_name VARCHAR(50),
    group_description VARCHAR(250),
    FOREIGN KEY (event_id) REFERENCES events(id) ON DELETE CASCADE
);

CREATE TABLE group_participants (
    user_id INT NOT NULL,
    group_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (group_id) REFERENCES groups(id) ON DELETE CASCADE
);

CREATE TABLE content (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    type INT NOT NULL,
    content BLOB NOT NULL,
    post_time DATETIME NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE event_content (
    id INT PRIMARY KEY AUTO_INCREMENT,
    event_id INT NOT NULL,
    content_id INT NOT NULL,
    FOREIGN KEY (event_id) REFERENCES events(id) ON DELETE CASCADE,
    FOREIGN KEY (content_id) REFERENCES content(id) ON DELETE CASCADE
);

CREATE TABLE group_content (
    id INT PRIMARY KEY AUTO_INCREMENT,
    group_id INT NOT NULL,
    content_id INT NOT NULL,
    FOREIGN KEY (group_id) REFERENCES groups(id) ON DELETE CASCADE,
    FOREIGN KEY (content_id) REFERENCES content(id) ON DELETE CASCADE
);

CREATE TABLE content_comments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    content_id INT NOT NULL,
    user_id INT NOT NULL,
    comment_text VARCHAR(250) NOT NULL,
    post_time DATETIME,
    FOREIGN KEY (content_id) REFERENCES content(id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

CREATE TABLE requests (
    id int PRIMARY KEY AUTO_INCREMENT,
    source_id INT NOT NULL,
    dest_id INT NOT NULL,
    group_id INT NOT NULL,
    request_type VARCHAR(7) NOT NULL,
    request_status BOOLEAN NOT NULL,
```

```sql
    FOREIGN KEY (source_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (dest_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (group_id) REFERENCES groups(id) ON DELETE CASCADE
);

CREATE TABLE messages (
    id INT PRIMARY KEY AUTO_INCREMENT,
    source_id INT NOT NULL,
    dest_id INT NOT NULL,
    message_text VARCHAR(250) NOT NULL,
    sent_time DATETIME NOT NULL,
    FOREIGN KEY (source_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (dest_id) REFERENCES users(id) ON DELETE CASCADE
);
```

25.2 Populating the Database

```
#System Config

INSERT INTO system_config (charge_rate) VALUES

(1.00),

(10.00),

(100.00),

(1000.00),

(10000.00),

(100000.00),

(1000000.00);


#Roles

INSERT INTO roles (user_role, auth_lvl) VALUES

('sysadmin', 0),

('controller', 1),

('event_manager', 2),

('group_manager', 3),

('participant', 4);


#Users

INSERT INTO users (role_id, username, user_password, email, first_name, last_name,
adr_number, street, city, dob) VALUES


#sysadmin

(1, 'pepe','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email0@service.com', 'Pepe',
'Pepeson', '1', 'Straight Lane', 'Someville', '1990-11-11'),


#controller

(2,'whoareu','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email1@service.com',
'Alice', 'Alison', '1', 'Straight Lane', 'Someville', '1990-11-11'),


#event managers

(3, 'guile','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email2@service.com', 'Mark',
'Markson', '1', 'Straight Lane', 'Someville', '1990-11-11'),
```

```
(3, 'akuma','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email3@service.com', 'Luke',
'Lukeson', '1', 'Straight Lane', 'Someville', '1990-11-11'),


#group managers

(4, 'kage','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email4@service.com', 'Jerri',
'Jerrison', '1', 'Straight Lane', 'Someville', '1990-11-11'),

(4, 'axl','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email5@service.com', 'Booker',
'Bookerson', '1', 'Straight Lane', 'Someville','1990-11-11'),

(4, 'el gado','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email6@service.com',
'Carlton', 'Carltonson', '1', 'Straight Lane', 'Someville','1990-11-11'),

(4, 'eden','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email7@service.com', 'Carl',
'Carlson', '1', 'Straight Lane', 'Someville', '1990-11-11'),


#users

(5, 'celeste','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email8@service.com',
'Conor', 'Conorson', '1', 'Straight Lane', 'Someville', '1990-11-11'),

(5, 'geki','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email9@service.com', 'Loren',
'Lorenson', '1', 'Straight Lane', 'Someville', '1990-11-11'),

(5, 'gill','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email10@service.com',
'Alecia', 'Aleciason', '1', 'Straight Lane', 'Someville', '1990-11-11'),

(5, 'hakan','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email11@service.com',
'Hugh', 'Hughson', '1', 'Straight Lane', 'Someville', '1990-11-11'),

(5, 'hokuto','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email12@service.com',
'Brant', 'Brantson', '1', 'Straight Lane', 'Someville', '1990-11-11'),

(5, 'cammy','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email13@service.com',
'Hiram', 'Hiramson', '1', 'Straight Lane', 'Someville', '1990-11-11'),

(5, 'ryu','5d1891fc9e394f5b79aada7d6ad8f7dda49aba5f', 'email14@service.com', 'Haley',
'Haleson', '1', 'Straight Lane', 'Someville', '1990-11-11');


#Debit details for event managers

INSERT INTO debit_details (user_id, bank_num, account_code ) VALUES

(3, 123456, 12345678),

(4, 987654, 98765432);


#Organizations for event managers

INSERT INTO organizations (user_id, org_name, org_type ) VALUES

(3, "Organization 1", "Family"),

(4, "Organization 2", "Corporation");
```

```
#Events
INSERT INTO events (manager_id, event_name, event_status, fee, event_description,
lifetime) VALUES
(3, 'Event 1', true, 10, 'The first event', '2019-11-30 16:00:00'),
(3, 'Event 2', true, 10, 'The second event', '2019-11-30 16:00:00'),
(3, 'Event 3', true, 10, 'The third event', '2019-11-30 16:00:00'),
(4, 'Event 4', true, 10, 'The fourth event', '2019-11-30 16:00:00'),
(4, 'Event 5', true, 10, 'The fifth event', '2019-11-30 16:00:00'),
(4, 'Event 6', true, 10, 'The sixth event', '2019-11-30 16:00:00');

#Groups
INSERT INTO groups (event_id, manager_id, group_name, group_description) VALUES
(1, 5, 'Group 1', 'The first group'),
(1, 6, 'Group 2', 'The second group'),
(2, 7, 'Group 3', 'The third group'),
(3, 8, 'Group 4', 'The fourth group'),
(4, 5, 'Group 5', 'The fifth group'),
(4, 6, 'Group 6', 'The sixth group'),
(5, 7, 'Group 7', 'The seventh group'),
(6, 8, 'Group 8', 'The eigth group'),
(6, 8, 'Group 9', 'The ninth group'),
(6, 8, 'Group 10', 'The tenth group');

#Event Participants
INSERT INTO event_participants (user_id, event_id) VALUES
(9,1),
(10,1),
(11,1),
(12,1),
(5,1),
(6,1),
(10,2),
(7,2),
```

```
(9,3),
(8,3),
(10,4),
(11,4),
(12,4),
(5,4),
(6,4),
(7,5),
(8,6);

#Group Participants
INSERT INTO group_participants (user_id, group_id) VALUES
(9,1),
(10,1),
(11,1),
(12,2),
(12,1),
(10,3),
(9,4),
(10,5),
(11,6),
(12,6),
(5,1),
(6,2),
(7,3),
(8,4),
(5,5),
(6,6),
(7,7),
(8,8);
```

```
#Insert content with random dates

INSERT INTO content (user_id, type, content, post_time) VALUES

(9,0, "Lorem ipsum dolor sit amet, consectetur adipiscing elit.", CURRENT_TIMESTAMP -
INTERVAL FLOOR(RAND() * 14) DAY),

(10,0, "Nulla fringilla a ante posuere ornare.", CURRENT_TIMESTAMP - INTERVAL
FLOOR(RAND() * 14) DAY),

(11,0, "Vivamus malesuada vitae quam ullamcorper ultrices. Phasellus tempor metus sed
pharetra commodo.", CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() * 14) DAY),

(12,0, "Duis viverra nulla tellus, vitae lobortis enim ultrices in. Mauris pulvinar
consectetur ipsum, ut rutrum massa condimentum et.", CURRENT_TIMESTAMP - INTERVAL
FLOOR(RAND() * 14) DAY),

(10,0, "Vestibulum dignissim eget nisi venenatis blandit.", CURRENT_TIMESTAMP -
INTERVAL FLOOR(RAND() * 14) DAY),

(6,0, "In leo felis, placerat sit amet massa et, sagittis volutpat tortor.",
CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() * 14) DAY),

(7,0, "Morbi et quam lectus.", CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() * 14) DAY),

(8,0, "Pellentesque aliquam nec tortor sit amet tincidunt.", CURRENT_TIMESTAMP -
INTERVAL FLOOR(RAND() * 14) DAY),

(5,0, "Cras ultricies lectus vel massa molestie, et molestie odio ornare.",
CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() * 14) DAY),

(6,0, "Nullam non dolor dignissim, convallis lorem in, congue purus.",
CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() * 14) DAY),

(7,0, "Donec ut urna volutpat, facilisis urna eget, accumsan odio.",
CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() * 14) DAY),

(10,0, "Donec finibus lobortis lacinia.", CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() *
14) DAY),

(11,0, "Nam id mauris est.", CURRENT_TIMESTAMP - INTERVAL FLOOR(RAND() * 14) DAY);


#Linking content to groups

INSERT INTO group_content (group_id, content_id) VALUES

(1,1),

(2,4),

(3,2),

(4,8),

(5,9),

(3,11),

(6,13);
```

```
#Linking content to events

INSERT INTO event_content (event_id, content_id) VALUES

(1,3),

(1,5),

(4,6),

(2,7),

(1,10),

(5,11);


#Comments

INSERT INTO content_comments (content_id, user_id, comment_text, post_time) VALUES

(1,5,"Donec hendrerit blandit leo, quis cursus augue vulputate a.",
CURRENT_TIMESTAMP),

(1,10, "Maecenas sit amet hendrerit orci.", CURRENT_TIMESTAMP + INTERVAL 1 HOUR),

(1,11, "Phasellus dignissim arcu id efficitur luctus.", CURRENT_TIMESTAMP + INTERVAL
2 HOUR),

(2,7, "Suspendisse non pellentesque enim.", CURRENT_TIMESTAMP),

(3,5, "Vivamus a arcu nec mauris auctor tristique.", CURRENT_TIMESTAMP),

(5,12, "Duis rutrum quis risus pulvinar facilisis.", CURRENT_TIMESTAMP),

(6,6, "Vivamus ut egestas ligula, non lobortis velit.", CURRENT_TIMESTAMP),

(6,11, "Donec bibendum ut mauris vel ullamcorper.", CURRENT_TIMESTAMP+ INTERVAL 2
HOUR),

(4,6, "Nullam euismod cursus dui, eget aliquet mi accumsan ac.", CURRENT_TIMESTAMP),

(8,8, "Proin rutrum diam quis bibendum tempor.", CURRENT_TIMESTAMP);


#Messages

INSERT INTO messages (source_id, dest_id, message_text, sent_time) VALUES

(10,1, "Nulla faucibus felis ligula, quis vulputate justo iaculis at.",
CURRENT_TIMESTAMP),

(11,1, "Pellentesque scelerisque, leo eget condimentum vulputate, turpis velit
venenatis sem, quis vehicula ante magna in nulla.", CURRENT_TIMESTAMP + INTERVAL 1
HOUR),

(12,1, "Morbi eget iaculis risus.", CURRENT_TIMESTAMP + INTERVAL 2 HOUR),

(1,7, "Suspendisse potenti. Proin hendrerit ipsum sed erat vehicula scelerisque.",
CURRENT_TIMESTAMP + INTERVAL 3 HOUR),

(2,6, "Nulla tincidunt blandit dui at fermentum.", CURRENT_TIMESTAMP),
```

```
(3,5, "Vestibulum imperdiet nisl ac est sollicitudin, et tincidunt mi congue.",
CURRENT_TIMESTAMP),

(4,4, "Vivamus vel ex tristique, pretium felis vitae, tempor felis.",
CURRENT_TIMESTAMP),

(5,3, "Nullam suscipit lobortis blandit. ", CURRENT_TIMESTAMP),

(6,2, "Phasellus vel ex sed erat ultrices auctor.", CURRENT_TIMESTAMP),

(7,8, "In iaculis dolor ante, vitae bibendum lectus ultrices tincidunt.",
CURRENT_TIMESTAMP);


#Requests

INSERT INTO requests (source_id, dest_id, group_id, request_type, request_status)
VALUES

(5,6,1, "invite",0),

(6,9,2, "invite",0),

(7,11,3, "invite",0),

(8,13,4, "invite",0),

(8,14,4, "invite",0),

(15,8,4, "join",0),

(14,7,3, "join",0),

(13,6,2, "join",0),

(15,6,2, "join",0),

(11,6,2, "join",0);
```

26 References

[1]: StatCounter. (2019). Desktop Browser Market Share Worldwide. Retrieved from URL: https://gs.statcounter.com/browser-market-share/desktop/worldwide

[2]: N.a, N.d. Data-Centered Architecture. *Tutorialspoint.* Retrieved from URL: tutorialspoint.com/software_architecture_design/data_centered_architecture.html

[3]: Reenskaug, Trygve, Coplien, James O. (20 Mar 2009) Data-Centered Architecture. "The DCI Architecture: A New Vision of Object-Oriented Programming". *Artima Developer*. Retrieved from URL: https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci_vision.html

[4]: Desai, Bipin C. (2019). *Relational Model & Relational Database Design, PDF lecture notes*. Retrieved from URL: https://confsys.encs.concordia.ca/CrsMgr/local_upload/1610/Teaching/COMP%20353/CM/1493/5-Reldsgn.pdf

[5]: N.a. N.d. Relational Database Design (RDD. In *Techopedia*. Retrieved from URL: https://www.techopedia.com/definition/25113/relational-database-design-rdd