



Final Project - Group 29
COMP353 – Databases

Mair Elbaz, 40004558
Daniel Vigny-Pau, 40034769
Francois David, 40046319
Alexandre Therrien, 40057134
Charles-Antoine Guite, 40063098

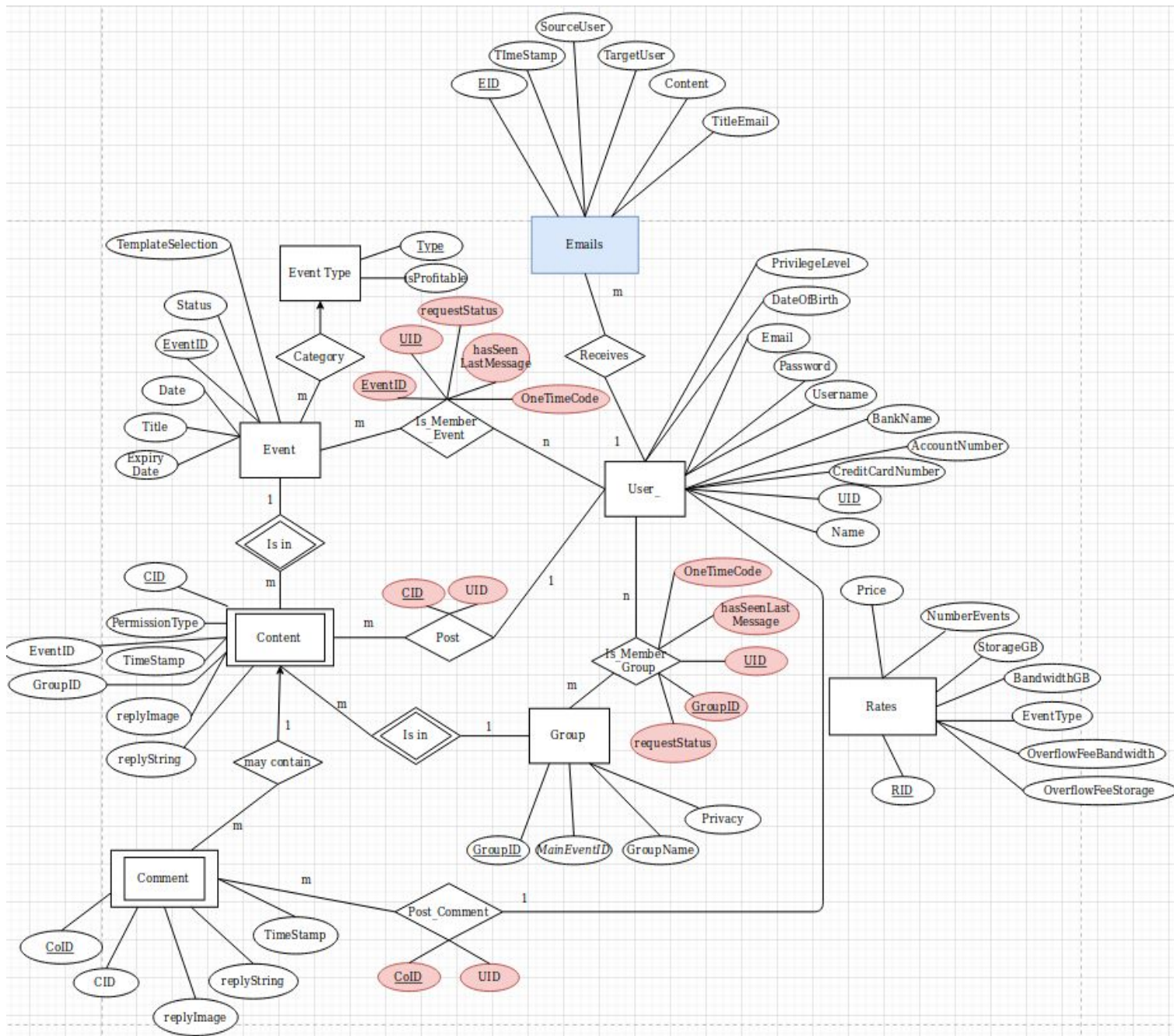
Professor: Dr. Desai

Fall 2019
Concordia University

CONTENTS:

| | |
|---|-----------|
| 1. ER Diagram..... | 3 |
| 2. Creating the tables..... | 4 |
| 3. Extracting data from the database..... | 9 |
| 4. Adding and modifying data from the database..... | 17 |
| 5. Deleting and removing data from the database..... | 28 |
| 6. Specific test cases..... | 35 |
| 7. Sample session & UI guide | 43 |
| 8. Possible scenarios..... | 69 |
| 9. Installation guide..... | 71 |
| 10. ReadMe..... | 73 |
| 11. Contributions..... | 75 |

1. ER diagram (3NF)



Explanation of the ER Diagram

User

The User entity stores all the debit information of the user, along with his/her credentials and other required fields for the User to have all that he/she needs to become an event manager. The only field worth explaining is the PrivilegeLevel field: it has a value of either 0, 1 or 2. A value of 0 is for a regular user, a value of 1 is for a controller, and a value of 2 is for an administrator.

Emails

The Emails entity stores the source and target user variables, which are foreign keys for the table User_ on UID. A timestamp field is used here to order the emails once they are put in the inbox of a user. The first emails should be the most recent ones. TitleEmail and Content are respectively for the title of the email and the content of the email. EID is the primary key. This entity is related to the User entity because the user has emails which belong/ are target towards him/her.

Is_Member_Group

This entity stores a variable called hasSeenLastMessage, which determines whether a user has seen the latest reply, a requestStatus variable, which determines whether a user is 'pending', 'member', or 'admin'. In the case the user is pending, this would mean that a request has been sent either to a user, or a request has been sent to the administrator of the event/group to authorize the person to come in. To know which case it is, we check the variable OneTimeCode: If it is set to something else than null, it means the administrator wants a user to join the event. Otherwise, it is the user that wants to join the group. The OneTimeCode is a random number given to a user so he/she can register to the event only using that code. Both the GroupID (which is the ID of the Group) and the UID (the ID of the user) are the primary keys in this table. They are foreign keys to the tables Group and User respectively. This is because the relationship

between Group and User is many to many, as a User can go in many groups and a Group is composed of many members.

Group

The GroupID is the ID representing the group. The MainEventID is the ID of the event in which the Group belongs. This is why at all times the Group must belong in an event. Without an event, the group cannot exist. This is why if an event gets removed by an administrator of the event, the groups also disappear. The GroupName, which is self explanatory. Finally, the Privacy, which is a tiny int that serves the purpose of telling us whether or not this group can be seen by all the members in the event. By default, a group is private, which means no one can see it actually exist except the members in it. Otherwise, it is public, and anyone from the event can join under authorization of the group administrator of course.

Is_Member_Event

This table serves the same purpose as does the Is_Member_Group table. The only difference here is that it concerns events. It contains two primary keys which are foreign keys to the table Event and table User. See the explanation of the variables above in the Is_Member_Group section.

Event

This table stores all the events in the program. It contains an ExpiryDate and a Date. The ExpiryDate determines if the event should be archived or not. The Date determines when does the event take place. The Title variable is the name of the event. The Status variable determines if the event has been approved by the administrator. If not, the event will not show even though the user paid for it. The event will show once the administrator approves it. TemplateSelection determines which template will be chosen to display the event. Also, notice that we did not put any variable named isArchived as we originally wanted to put: This is because the expiryDate can answer that question, which would not have made our

tables 3NF. The Event also contains a foreign key to an Event_Type and to a Rate.

Event_Type

This table has been separated from the Event table to store less redundant information and to keep our ER diagram in 3NF. This is because once we know the type of the event, we can tell whether it is profitable or not. So, a separate table has been created to store a VARCHAR that tells us what is the name for the type of event and a tiny int named isProfitable which determines if the user will have to pay to create this type of event.

Content

This table stores all the content of the software. This means every post that is made goes through the content table. Even the comments, although they go in the Comments table, because they rely on the ID of the Content. The EventID and GroupID (foreign keys to tables Event and Group) variables determine whether this content belongs to an event or a group. It is still 3NF because of the way we use them: EventID is always set, and GroupID is only set when the Content goes into a Group. The PermissionType variable determines whether a post cannot receive comments (0), can receive comments (1), and can receive comments with links (2). The TimeStamp determines when the Content was posted: this becomes very useful to put the Content into the order they were written. ReplyImage is the image, if any, that the user sent with the content, and replyString is the words that have been sent as the content. CID is the primary key.

Comment

The comment table is populated with the replies to the posts that are made on the event discussion. Only the event administrator can post content, and then the users reply to that content inside of comments. The comments contain a replyString, a replyImage, and a TimeStamp value, which are just like those values in the Content table. The only difference here is that we have a foreign key to the CID attribute of the Content table, which tells us

under which post does the Comment goes. We do not want 'Hey user!' to go under a content like 'There is hail outside', we want it to go where it belongs, under a post like 'Hey all members!'.

Post and Post_Comment

These tables are also the same, apart from the fact that one directs to the Comments table and the other directs to the Content table. These tables determine who posted some content. It is very easy to find out who posted some information, as the ID of the Content/Comment is one attribute, and the UID of the User is the other. We can easily get the username by looking at whom the UID corresponds. Although this table might not be necessary, it is a very fast method to extract the information we need.

Rates

Although not shown in the ER diagram, this table is linked to the Event table. Lack of space is the reason why we do not see them linked. The reason it is linked to the Event is because the user, when paying for an event, selects to pay for a certain package. This package, or rate, selected is now related by foreign key to the event so that whenever we need to check how much the user needs to pay we can simply go find the event and know the Rate he/she needs to pay.

Storage GB is the storage in GB that comes with the rates. Bandwidth GB is the bandwidth allowed in GB that comes with the rates.

OverflowFeeBandwidth is the price paid for every GB the user goes over the maximum bandwidth he/she was allowed. OverflowFeeStorage is the price paid for every GB the user goes over the maximum storage he/she was allowed. Price is the price paid for the rate (the package) selected with the event. NumberEvents determines if a user is allowed for a certain rates depending on the number of events he/she has created before.

Reasons for our design:

- Post and Post_Comment tables are used for consistency (they are consistent with the relations to the Event and Group tables, where intermediaries are required) and to access information fast.
- Event_Type is generated to keep our database structure in 3NF, as the name of the EventType makes it either profitable or not.
- Is_Member_Event and Is_Member_Group are the tables that are required to make the m to n relationship work. The easiest way to handle m to n relationships is as we did; by adding a table in between their relationship, and putting a composite primary key to the IDs of both tables.
- Comment table is added to show the relationship parent/child between the posts. The Content element has some child elements which are Comments.

File/Code Structure

Although on the server all the code is in one directory, it has been separated into many. One folder serves all the requests that we need to serve us with backend information for AJAX purposes (as we are using AJAX in our project to dynamically load information without needing to refresh the page). This makes it easy to see that the php files inside that directory do not serve html content. The css is all put in other files, and some javascript has been separated for easier management of files.

For the code structure, a database layer has been added so that we do not need to handle the database ourselves with mysql. What this means is that many functions have been created for the different cases there might be. In section 3, the code is seen in more details.

Input validation has been integrated in most if not all possible locations. A message is returned from the AJAX requests for the different cases a request may fail.

Security

The passwords are stored using PASSWORD_BCRYPT encryption, with the use of the password_hash function.

Also, URL manual manipulation attacks are also blocked, because any attempt for an unregistered user to go to another page link than the log in or register page will directly redirect them to the log in page. Also, user verification is also made before accessing the adminHome.php and controllerHome.php files to make sure only the admin and only the controller can access the page.

Finally, SQL injection attacks are mitigated by the use of prepared statements. The queries recognize that a query is composed of user input and will cancel all malicious attempts to gather information.

Limitations

We depend, as required, on an API for our project. The API do not always respond correctly to our requests. This might be because of the API key that we are given that stopped working for some reason, or some other unknown criteria we are not aware about.

2. Creating the tables

Creating the **Event_Type** table:

```
CREATE TABLE Event_Type (
    EventType varchar(20) not null,
    isProfitable boolean not null,
    primary key(EventType)
);
```

Creating the **Event_** table:

```
CREATE TABLE Event_ (
    EventID int unsigned not null auto_increment,
    Status boolean not null,
    Date varchar(20) not null,
    Title varchar(20) unique not null,
    ExpiryDate varchar(20) not null,
```

```

        EventType varchar(20) not null,
        foreign key (EventType) references Event_Type(EventType) on delete
        cascade,
        primary key (EventID)
    );

```

Creating the **Group_** table:

```

CREATE TABLE Group_ (
    GroupID int unsigned not null auto_increment,
    MainEventID int unsigned not null,
    GroupName varchar(20) unique not null,
    Privacy boolean not null,
    foreign key (MainEventID) references Event_(EventID) on delete
    Cascade,
    primary key (GroupID)
);

```

Creating the Content table:

```
CREATE TABLE Content (
    CID int unsigned not null auto_increment,
    PermissionType smallint not null,
    TimeStamp int(11) not null,
    replyImage blob,
    replyString varchar(500),
    EventID int unsigned not null,
    GroupID int unsigned, foreign key (EventID) references
    Event_(EventID) on delete cascade,
    foreign key (GroupID) references Group_(GroupID) on delete cascade,
    primary key (CID)
);
```

Creating the User_ table:

```
CREATE TABLE User_ (
    UID int unsigned not null auto_increment,
    Username varchar(20) unique not null,
    Password varchar(20) not null,
    Email varchar(30) not null,
    Name varchar(50) not null,
    DateOfBirth varchar(20) not null,
    PrivilegeLevel smallint unsigned not null,
    BankName varchar(30),
    AccountNumber bigint unsigned,
    CreditCardNumber bigint unsigned,
    Address varchar(20),
    PhoneNumber varchar(20),
    primary key (UID)
);
```

Creating the **Is_Member_Group** table:

```
CREATE TABLE Is_Member_Group (
    GroupID int unsigned not null,
    UID int unsigned not null,
    requestStatus varchar(10),
    hasSeenLastMessage boolean not null,
    primary key (GroupID, UID),
    foreign key (GroupID) references Group_(GroupID) on delete cascade,
    foreign key (UID) references User_(UID) on delete cascade
);
```

This table determines which participants belong to which groups. *hasSeenLastMessage* is used to determine if the members have seen the last message. If not a member, it is set to false or 0.

Creating the **Emails** table:

```
CREATE TABLE Emails (
    EID int unsigned not null auto_increment,
    TimeStamp int(11) not null,
    SourceUser int unsigned not null,
    TargetUser int unsigned not null,
    Content varchar(1000) not null,
    TitleEmail varchar(30) not null,
    foreign key (SourceUser) references User_(UID) on delete cascade,
    foreign key (TargetUser) references User_(UID) on delete cascade,
    primary key(EID)
);
```

These are the emails of users.

Creating the **Is_Member_Event** table:

```
CREATE TABLE Is_Member_Event (
    EventID int unsigned not null,
    UID int unsigned not null,
    requestStatus varchar(10),
    hasSeenLastMessage boolean not null,
    primary key (EventID, UID),
    foreign key (EventID) references Event_(EventID) on delete cascade,
    foreign key (UID) references User_(UID) on delete cascade
);
```

This table determines which participants belong to which events. *hasSeenLastMessage* is used to determine if the members have seen the last message. If not a member, it is set to false or 0.

Creating the **Post** table:

```
CREATE TABLE Post (
    CID int unsigned not null,
    UID int unsigned not null,
    primary key (CID),
    foreign key (CID) references Content(CID) on delete cascade,
    foreign key (UID) references User_(UID) on delete cascade
);
```

This table determines who posted a certain content. Since there is always a person who posts content, the primary key is only CID, not (CID, UID).

Creating the **Comment** table:

```
CREATE TABLE Comment (
    CoID int unsigned not null auto_increment,
    CID int unsigned not null,
    TimeStamp int(11) not null,
    replyString varchar(500),
    foreign key (CID) references Content(CID) on delete cascade,
    primary key (CoID)
);
```

This table is used whenever comments are made on Event posts with permission type set to 0 (allows comments).

Creating the **Post_Comment** table:

```
CREATE TABLE Post_Comment (
    CoID int unsigned not null,
    UID int unsigned not null,
    foreign key (CoID) references Comment(CoID) on delete cascade,
    foreign key (UID) references User_(UID) on delete cascade,
    primary key (CoID)
);
```

This table determines who posted a certain comment. It must be different from

Creating the **Rates** table:

```
CREATE TABLE Rates (
    NumberEvents int unsigned not null,
    EventType varchar(20) not null,
    StorageGB int unsigned not null,
    BandwidthGB int unsigned not null,
    Price decimal not null, foreign key (EventType) references
    Event_Type(EventType) on delete cascade,
    primary key(NumberEvents,EventType, StorageGB, BandwidthGB)
);
```

This table determines the rates of an event, depending on the type of event and number of events the user has.

Added the following rows to Event_Type as the constant types of events:

```
INSERT INTO Event_Type values ('family', 1);
INSERT INTO Event_Type values ('community', 1);
INSERT INTO Event_Type values ('non-profit', 0);
INSERT INTO Event_Type values ('profit', 1);
```

3. Extracting data from the database

All functions are written in PHP, where:

- **\$mysqli** is used to connect to the database object
- **\$username** is the username of the user (string)
- **\$eventTitle** is the title of an event (must exist) (string)
- **\$groupName** is the name of a group (string)
- **\$UID** is the UID of a user (int)
- **\$eventTitle** is the title of an event (string)
- **\$CID** is the Content ID (int)

Get the list of emails that were destined to the user:

```
function showEmailsReceived($mysqli, $username){
    //find if the user really exists
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'showEmailsReceived: User with username '.$username.' does
            not exist';
    }

    $result2 = $mysqli->query("SELECT SourceUser, TitleEmail, Content FROM
        Emails WHERE TargetUser='".$first_row[0]'" ORDER BY TimeStamp
        DESC;");
    return $result2;
}
```


Get the list of emails that were sent by the user:

```
function showEmailsSent($mysqli, $username){
    //find if the user really exists
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='\".$username.\"';");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'showEmailsSent: User with username ' . $username . ' does not
            exist';
    }

    $result2 = $mysqli->query("SELECT TargetUser, TitleEmail, Content FROM
        Emails WHERE SourceUser= '\".$first_row[0].\"' ORDER BY TimeStamp
        DESC;");
    return $result2;
}
```

Get all the members of an event:

```
function getEventMembers($mysqli, $eventTitle){
    //find eventID
    $result = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='\".$eventTitle.\"';");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getEventMembers: Event with title ' . $eventTitle . ' does not
            exist.';
    }

    //get UIDs of the people that belong to the event
    $result2 = $mysqli->query("SELECT UID FROM Is_Member_Event WHERE
        EventID= '\".$first_row[0].\"' AND (requestStatus='member' OR
        requestStatus='admin')");
    return $result2;
}
```

Get the UIDs of all pending users who wish to join an event or that have been invited by the event manager but are not members yet.

```
function getEventPendingUsers($mysqli, $eventTitle){
    //find eventID
    $result = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='". $eventTitle. "'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getEventPendingUsers: Event with title ' . $eventTitle . ' does
            not exist.';
    }

    //get UIDs of the people that belong to the event
    $result2 = $mysqli->query("SELECT UID FROM Is_Member_Event WHERE
        EventID='". $first_row[0]. "' AND requestStatus='pending'");
    return $result2;
}
```

Get the UIDs of members of a group, including the admin:

```
function getGroupMembers($mysqli, $groupName){
    //find groupID
    $result = $mysqli->query("SELECT GroupID FROM Group_ WHERE
        GroupName='". $groupName. "'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getGroupMembers: Group with group name ' . $groupName . ' does
            not exist.';
    }

    //get UIDs of the people that belong to the group.
    $result2 = $mysqli->query("SELECT UID FROM Is_Member_Group WHERE
        GroupID='". $first_row[0]. "' AND (requestStatus='member' OR
            requestStatus='admin')");
    return $result2;
}
```

Get the UIDs of all pending users who wish to join a group or who have been invited by the group manager but are not members yet.

```
function getGroupPendingUsers($mysqli, $groupName){
    //find groupID
    $result = $mysqli->query("SELECT GroupID FROM Group_ WHERE
        GroupName='".$groupName."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getGroupPendingUsers: Group with group name '.$groupName.'
            does not exist.';
    }

    //get UIDs of the people that belong to the group.
    $result2 = $mysqli->query("SELECT UID FROM Is_Member_Group WHERE
        GroupID='".$first_row[0]."' AND requestStatus='pending'");
    return $result2;
}
```

Get the username of a user based on their UID, returns false if the UID does not exist:

```
function getUsername($mysqli, $UID){
    $result = $mysqli->query("SELECT Username FROM User_ WHERE
        UID='".$UID."'");
    $first_row = mysqli_fetch_row($result);
    return $first_row[0];
}
```

Get all content for an event: the content ID (CID) and the replyString of a post, along with the UID (user ID) of the person who made the post.

```
function getContentEvent($mysqli, $eventTitle){
    $result = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='". $eventTitle. "'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getContentEvent: Event with title ' . $eventTitle . ' does not
            exist';
    }

    $result2 = $mysqli->query("SELECT Content.CID, Content.replyString,
        Post.UID FROM Content INNER JOIN Post ON Content.CID=Post.CID WHERE
        EventID='". $first_row[0]. "' AND GroupID IS NULL ORDER BY TimeStamp
        DESC;");
    return $result2;
}
```

Get all comments for a certain content post, only for posts in events. Returns the string of the post and UID (user who made the post).

```
function getCommentsContent($mysqli, $CID){
    $result = $mysqli->query("SELECT Comment.replyString, Post_Comment.UID
        FROM Comment INNER JOIN Post_Comment ON
        Comment.CoID=Post_Comment.CoID WHERE CID='". $CID. "' ORDER BY TimeStamp
        ASC;");
    return $result;
}
```

Get all content for a certain group, ordered correctly. Returns the strings of the content and the UID 2(ID of user posting content).

```
function getContentGroup($mysqli, $groupName){
    //find the group
    $result = $mysqli->query("SELECT GroupID FROM Group_ WHERE
        GroupName='".$groupName."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getContentGroup: Group with name '.$groupName.' could not be
            found';
    }

    //now look through DB.
    $result2 = $mysqli->query("SELECT Content.replyString, Post.UID FROM
        Content INNER JOIN Post ON Content.CID=Post.CID WHERE
        GroupID='".$first_row[0]'" ORDER BY TimeStamp ASC;");
    return $result2;
}
```

Get all the events a user belongs to. Returns a table of events of the user. The user must exist.

```
function getEventsOfUser($mysqli, $username){
    //find userID (UID)
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getEventsOfUser: User with username '.$username.' was not
            found.';
    }

    //get all the titles of the events the user belongs to and the EventID
    of those
    $result2 = $mysqli->query("SELECT Event_.EventID, Event_.Title FROM
        Event_ INNER JOIN Is_Member_Event ON
```

```

        Is_Member_Event.EventID=Event_.EventID WHERE
        Is_Member_Event.UID=".$first_row[0].";");
    return $result2;
}

```

Get the latest post of an event.

```

function getLatestPostEvent($mysqli, $eventID){
    $result = $mysqli->query("SELECT replyString, MAX(TimeStamp) FROM
        Content WHERE EventID=".$eventID." AND GroupID IS NULL GROUP BY
        replyString;");
    return $result;
}

```

Get all the groups a user belongs to. Returns the GroupIDs and GroupNames of the user.

```

function getGroupsOfUser($mysqli, $username){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getGroupsOfUser: User with username '.$username.' was not
        found.';
    }

    $result2 = $mysqli->query("SELECT Group_.GroupID, Group_.GroupName
        FROM Group_ INNER JOIN Is_Member_Group ON
        Is_Member_Group.GroupID=Group_.GroupID WHERE
        Is_Member_Group.UID=".$first_row[0].";");
    return $result2;
}

```

Get the latest post of a group. Returns the replyString that has the greatest timestamp value (the most recent post).

```
function getLatestPostGroup($mysqli, $groupID){
    $result = $mysqli->query("SELECT replyString, MAX(TimeStamp) FROM
        Content WHERE GroupID='".$groupID.'" GROUP BY replyString;");
    return $result;
}
```

Get all user info based on their username. Returns all the user details in a 1D array that can be referred to easily, such as \$first_row[0], etc.

```
function getUser($mysqli, $username){
    $result = $mysqli->query("SELECT * FROM User_ WHERE
        Username='".$username.'"");
    $first_row = mysqli_fetch_row($result);
    return $first_row;
}
```

Get all the groups that belong to an event. Returns all the group names that belong to an event.

```
function getGroupsInEvent($mysqli, $eventTitle){
    //first find event
    $result = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='".$eventTitle.'"");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'getGroupsInEvent: Event with title '.$eventTitle.' was not
            found.';
    }

    //now find all groups
    $result2 = $mysqli->query("SELECT GroupName FROM Group_ WHERE
        MainEventID='".$first_row[0].'"");
    return $result2;
}
```

NOTE: Queries have been modified to prepared statements to protect from SQL injection.

4. Adding and modifying data from the database

All functions are written in PHP, where:

- **\$mysqli** is used to connect to the database object
- **\$username** is the username of the user (string)
- **\$eventTitle** is the title of an event (must exist) (string)
- **\$groupName** is the name of a group (string)
- **\$UID** is the UID of a user (int)
- **\$eventTitle** is the title of an event (string)
- **\$CID** is the Content ID (int)
- **\$replyString** is the text (string) contained in a comment (can be empty) (string)
- **\$replyStringCID** is the text for the content being commented on (string)
- **\$replyImage** is the image contained in a comment (can be empty) (string)
- **\$sourceUsername** is the username of the sender of an email (string)
- **\$targetUsername** is the username of the receiver of an email (string)
- **\$date** is a date object of form 'YYYY-MM-DD'
- **\$title** is the name of an event (string)
- **\$expiryDate** is a date object of form 'YYYY-MM-DD'
- **\$eventType** is the type of an event (string)
- **\$usernameCreator** is the username of whoever created the event (string)
- **\$isProfitable** is a boolean determining if an event is profitable, 0 for no, 1 for yes (int)
- **\$numberEvents** is a positive integer that says how many events for a price (int)
- **\$storageGB** is the storage space the event takes in GB (int)
- **\$bandwidthGB** is the bandwidth an event takes in GB (int)
- **\$price** is the price for a package (double)
- **\$password** is the password of a user (string)
- **\$email** is the email address of a user (string)
- **\$name** is the full name of a user (first and last) (string)
- **\$dateofbirth** is the birth date of a user in format 'YYYY-MM-DD'
- **\$privilegelevel** 0 is standard privilege, 1 is for controller, 2 is for admin (int)

Create a new comment (from a post in an event).

```
function addComment($mysqli, $replyString, $replyStringCID, $username){
    //search for the content made with timestamp timestampCID
    $result = $mysqli->query("SELECT CID,PermissionType,GroupID FROM
        Content WHERE replyString='".$replyStringCID."'");
    $first_row = mysqli_fetch_row($result);
```



```

if(is_bool($first_row[0])){
    return 'addComment: There is no such content which has this string
    as content: '.$replyStringCID;
}

//get user and see if exists
$result2 = $mysqli->query("SELECT UID FROM User_ WHERE
    Username='".$username."'");
$first_row_2 = mysqli_fetch_row($result2);
if(is_bool($first_row_2[0])){
    return 'addComment: There is no such user that has this username:
    '.$username;
}

if($first_row[2]!=''){
    return 'addComment: Content comes from a group. Cannot comment on
    some content from a group.';
}

if($first_row[1]==0){
    return 'addComment: Content cannot be commented due to privilege
    level';
}

if($first_row[1]==1){
    //only comment is accepted. No links!
    if(strpos($replyString, "www.") !== false){
        return 'addComment: There is a link inside the comment; according
        to the privilege level, cannot be sent.';
    }
}

$timestamp = time();
$mysqli->query("INSERT INTO Comment (CID, replyString, TimeStamp)
    VALUES (". $first_row[0].", '". $replyString."', '". $timestamp."')");
//find CoID of just added entity
$result3 = $mysqli->query("SELECT CoID FROM Comment WHERE
    TimeStamp='".$timestamp."'");
$first_row_3 = mysqli_fetch_row($result3);

```

```

$mysqli->query("INSERT INTO Post_Comment (CoID, UID) VALUES
    (". $first_row_3[0].", ". $first_row_2[0].");");
return 'addComment: ' . $mysqli->error;
}

```

Create new content (reply in a group, post in an event).

```

function addContent($mysqli, $permissionType, $replyImage, $replyString,
    $seventTitle, $groupName, $username){
    //find event related
    $result = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='". $seventTitle."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'addContent: Could not find any Event with the name
            ' . $seventTitle . "<br>";
    }

    //find group related (if any)
    $result2 = 0;
    $first_row_2 = [FALSE];
    if($groupName!=''){
        $result2 = $mysqli->query("SELECT GroupID FROM Group_ WHERE
            GroupName='". $groupName."'");
        $first_row_2 = mysqli_fetch_row($result2);
        if(is_bool($first_row_2[0])){
            return 'addContent: Could not find any Group with the name
                ' . $groupName . "<br>";
        }
    }

    //find the user that posted this content
    $result3 = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='". $username."'");
    $first_row_3 = mysqli_fetch_row($result3);

    if(is_bool($first_row_3[0])){

```

```

    return 'addContent: Could not find any user with username
    '.$username."<br>";
}

if($permissionType>2 || $permissionType<0){
    return 'addContent: Error: Invalid permission type';
}

//get current timestamp
$timestamp = time();

//add to content
if(is_bool($first_row_2[0])){
    if($replyImage==''){
        $mysqli->query("INSERT INTO Content (EventID, PermissionType,
        TimeStamp, replyString) VALUES
        (\".$first_row[0].\", \".$permissionType.\", \".$timestamp.\", '\".$replyStrin
        g.\"');");
    } else{
        //not working
        $mysqli->query("INSERT INTO Content (EventID, PermissionType,
        TimeStamp, replyImage, replyString) VALUES
        (\".$first_row[0].\", \".$permissionType.\", \".$timestamp.\", \".$file_get_con
        tents($replyImage).\", '\".$replyString.\"');");
    }
} else{
    if($replyImage==''){
        $mysqli->query("INSERT INTO Content (EventID, GroupID,
        PermissionType, TimeStamp, replyString) VALUES
        (\".$first_row[0].\", \".$first_row_2[0].\", 0, \".$timestamp.\", '\".$replyStr
        ing.\"');");
    } else{
        //not working
        $mysqli->query("INSERT INTO Content (EventID, GroupID,
        PermissionType, TimeStamp, replyImage, replyString) VALUES
        (\".$first_row[0].\", \".$first_row_2[0].\", 0, \".$timestamp.\", \".$file_get_c

```

```

        contents($replyImage).", ' ".$replyString."'");");
    }
}
$val=$mysqli->query("SELECT CID FROM Content WHERE
    TimeStamp=".$timestamp.");");
$first_row_4 = mysqli_fetch_row($val);
if(!is_bool($first_row_4[0])){
    $mysqli->query("INSERT INTO Post (CID, UID) VALUES
        ( ".$first_row_4[0].", ".$first_row_3[0].");");
}
return 'addContent: '.$mysqli->error;
}

```

Create new email message.

```

function sendEmail($mysqli, $sourceUsername, $targetUsername, $titleEmail,
    $replyString){
    //find source user and target user
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$sourceUsername."'");
    $first_row = mysqli_fetch_row($result);
    if(is_bool($first_row[0])){
        return 'sendEmail: Source username does not exist';
    }

    $result2 = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$targetUsername."'");
    $first_row_2 = mysqli_fetch_row($result2);
    if(is_bool($first_row_2[0])){
        return 'sendEmail: Target username does not exist';
    }

    $timestamp = time();

    //now add to email list
    $mysqli->query("INSERT INTO Emails (TimeStamp, SourceUser, TargetUser,
        TitleEmail, Content) VALUES
        ( ".$timestamp.", ".$first_row[0].", ".$first_row_2[0].", ".$titleEmail

```

```

        ."', '". $replyString.'');");
    return 'sendEmail: ' . $mysqli->error;
}

```

Create new event:

```

function createEvent($mysqli, $date, $title, $expiryDate, $eventType,
$usernameCreator){
    if(doesEventTypeExist($eventType)){
        return 'createEvent: Was this event type added to the event types
table?';
    }

    if($date[4] != '-' || $date[7] != '-' || strlen($date) != 10){
        return 'createEvent: Wrong date format';
    }
    if($expiryDate[4] != '-' || $expiryDate[7] != '-' ||
strlen($expiryDate) != 10){
        return 'createEvent: Wrong expiryDate format';
    }
    //make sure expiryDate > date! Otherwise event will be archived

    //make sure the creator actually exists, and make him the admin of the
event.
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
Username='". $usernameCreator."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'createEvent: The username provided does not correspond to
any user';
    }

    $mysqli->query("INSERT INTO Event_ (Status, Date, Title, ExpiryDate,
EventType) values
(1, '". $date.'', '". $title.'', '". $expiryDate.'', '". $eventType.'')");
    $error = $mysqli->error;
}

```

```

        //get the event id just created
        $result2 = $mysqli->query("SELECT EventID FROM Event_ WHERE
Date='".$date.'" AND Title='".$title.'";");
        $first_row_2 = mysqli_fetch_row($result2);

        //no need to verify if insertion worked; I just added the event

        //now add the user to the event
        //this tells that the user is the admin and that he has seen the
        latest message (event not created, nothing in it)
        $mysqli->query("INSERT INTO Is_Member_Event (EventID, UID,
requestStatus, hasSeenLastMessage) VALUES
('".$first_row_2[0]."', '".$first_row[0]."', 'admin', 1)");

        return 'createEvent: '.$error;
    }

```

Add an event type.

```

function addEventType($mysqli, $eventType, $isProfitable){
    $mysqli->query("INSERT INTO Event_Type values
        ('".$eventType."', '".$isProfitable.'");");
    return 'addEventType: '.$mysqli->error;
}

```

Create a new group.

```

function createGroup($mysqli, $eventTitle, $groupName, $usernameCreator){
    //first search for event
    $result = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='".$eventTitle.'";");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'createGroup: Event with title '.$eventTitle.' does not
        exist';
    }

    //then search for user creator. Need to verify the user is already a

```

```

    member of the event!
$result2 = $mysqli->query("SELECT UID FROM User_ WHERE
    Username='".$usernameCreator."'");
$first_row_2 = mysqli_fetch_row($result2);

if(is_bool($first_row_2[0])){
    return 'createGroup: User with username '.$usernameCreator.' does
    not exist';
}

//because only a user in the event can create a group...
$findUserInEvent = $mysqli->query("SELECT requestStatus FROM
    Is_Member_Event WHERE EventID='".$first_row[0]."' AND
    UID='".$first_row_2[0]."'");
$first_row_3 = mysqli_fetch_row($findUserInEvent);

if(is_bool($first_row_3[0])){
    return 'createGroup: User with username '.$usernameCreator.' cannot
    create a group because he has sent a demand to join the event that
    has not been answered or is simply not part of the group.';
}

//then add group with privacy 0 (group is private, so no showing)
$mysqli->query("INSERT INTO Group_ (MainEventID, GroupName, Privacy)
    values ('.intval($first_row[0]).','.$groupName.',0);");
$error=$mysqli->error;

//find our new group ID
$result3 = $mysqli->query("SELECT GroupID FROM Group_ WHERE
    GroupName=''.$groupName.'");
$first_row_3 = mysqli_fetch_row($result3);

//then add user as admin of group. hasSeenLastMessage=0 means he has
    seen the last message of the group (since there is nothing...).
$mysqli->query("INSERT INTO Is_Member_Group (GroupID, UID,
    requestStatus, hasSeenLastMessage) VALUES
    ('.$first_row_3[0].','.$first_row_2[0].','admin',1)");

```

```

    return 'createGroup: '.$error;
}

```

Add a member to an event, only to be used by an Event Manager.

```

function addUserToEvent($mysqli, $username, $eventTitle){
    //find member with that username
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);
    //find event with that title
    $result2 = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='".$eventTitle."'");
    $first_row_2 = mysqli_fetch_row($result2);
    //now create the entry. set user with status 'pending'
    $mysqli->query("INSERT INTO Is_Member_Event (EventID, UID,
        requestStatus, hasSeenLastMessage) values
        (".intval($first_row_2[0]).", ".intval($first_row[0]).", 'pending', 0);
        ");
    return 'addUserToEvent: '.$mysqli->error;
}

```

Add a member to a group, only to be used by a Group Manager.

```

function addUserToGroup($mysqli, $username, $groupName, $eventTitle){
    //find member with that username
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    $result3 = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='".$eventTitle."'");
    $first_row_3 = mysqli_fetch_row($result3);

    //because only a user in the event can create a group...
    $findUserInEvent = $mysqli->query("SELECT UID, requestStatus FROM
        Is_Member_Event WHERE EventID=".$first_row_3[0]. " AND
        UID=".$first_row[0]. " ");
}

```



```

$first_row_2 = mysqli_fetch_row($findUserInEvent);
if(is_bool($first_row_2[0])){
    return 'addUserToGroup: User with username '.$username.' cannot join
    a group because he has sent a demand to join the event that has not
    been answered or is simply not part of the group.';
}

//find event with that title
$result2 = $mysqli->query("SELECT GroupID FROM Group_ WHERE
    GroupName='".$groupName."'");
$first_row_2 = mysqli_fetch_row($result2);
//now create the entry. set user with status 'pending'
$mysqli->query("INSERT INTO Is_Member_Group (GroupID, UID,
    requestStatus, hasSeenLastMessage) values
    (".intval($first_row_2[0]).", ".intval($first_row[0]).", 'pending', 0);
    ");
return 'addUserToGroup: '.$mysqli->error;
}

```

Add some new rates, only to be used by a Controller.

```

function addRates($mysqli, $numberEvents, $storageGB, $bandwidthGB,
    $eventType,
    $price){
    if(doesEventTypeExist($eventType)){
        return 'addRates: Was this event type added to the event types
        table?';
    }

    $mysqli->query("INSERT INTO Rates values
        (".$numberEvents.", '". $eventType."', ". $storageGB.", ". $bandwidthGB.",
        ". $price.");");
    return 'addRates: '.$mysqli->error;
}

```

Add a new user. Outputs possible errors.

```
function addUser($mysqli, $username, $password, $email, $name,
    $dateofbirth,
    $privilegelevel){
    //check date of birth format
    if($dateofbirth[4] != '-' || $dateofbirth[7] != '-' ||
        strlen($dateofbirth) != 10){
        return 'addUser: Wrong dateofbirth format';
    }
    //check privilege level
    if($privilegelevel > 2 || $privilegelevel < 0){
        return 'addUser: Wrong privilegelevel';
    }
    $mysqli->query("INSERT INTO User_ (Username, Password, Email, Name,
        DateOfBirth, PrivilegeLevel) values
        ('".$username."', '".$password."', '".$email."', '".$name."', '".$dateof
        birth."', ".$privilegelevel.");");
    return 'addUser: '.$mysqli->error;
}
```

Check if event type exists.

```
function doesEventTypeExist($eventType){
    return strcmp($eventType, 'family') != 0 && strcmp($eventType,
        'community') != 0 &&
        strcmp($eventType, 'non-profit') != 0 && strcmp($eventType,
        'profit') != 0;
}
```

NOTE: Queries have been modified to prepared statements to protect from SQL injection.

5. Deleting and removing data from the database

All functions are written in PHP, where:

- **\$mysqli** is used to connect to the database object
- **\$username** is the username of the user (string)
- **\$eventTitle** is the title of an event (must exist) (string)
- **\$groupName** is the name of a group (string)
- **\$UID** is the UID of a user (int)
- **\$eventTitle** is the title of an event (string)
- **\$CID** is the Content ID (int)

Remove a user from an event (for admin, event manager).

```
function removeUserFromEvent($mysqli, $username, $eventTitle){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'removeUserFromEvent: User with username '.$username.' was
        not found.';
    }

    $result2 = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='".$eventTitle."'");
    $first_row_2 = mysqli_fetch_row($result2);

    if(is_bool($first_row_2[0])){
        return 'removeUserFromEvent: Event with title '.$eventTitle.' was
        not found.';
    }

    //now remove the user with Is_Member_Event table
    $mysqli->query("DELETE FROM Is_Member_Event WHERE
        UID=".$first_row[0]." AND EventID=".$first_row_2[0].");
    //also remove all groups that the user belongs to in this event...
    $result3 = $mysqli->query("SELECT GroupID FROM Group_ WHERE
        MainEventID=".$first_row_2[0].");
```

```

//now for all of them, remove the user's membership to the group
while($row=mysqli_fetch_row($result3)){
    $mysqli->query("DELETE FROM Is_Member_Group WHERE
    UID=".$first_row[0]. " AND GroupID=".$row[0].";");
}
return 'removeUserFromEvent: '.$mysqli->error;
}

```

Remove a user from a group (for admin, group manager).

```

function removeUserFromGroup($mysqli, $username, $groupName){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
    Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'removeUserFromGroup: User with username '.$username.' was
        not found.';
    }

    $result2 = $mysqli->query("SELECT GroupID FROM Group_ WHERE
    GroupName='".$groupName."'");
    $first_row_2 = mysqli_fetch_row($result2);

    if(is_bool($first_row_2[0])){
        return 'removeUserFromGroup: Group with name '.$groupName.' was not
        found.';
    }

    $mysqli->query("DELETE FROM Is_Member_Group WHERE
    UID=".$first_row[0]. " AND GroupID=".$first_row_2[0].";");
    return 'removeUserFromGroup: '.$mysqli->error;
}

```

Remove a user from the system entirely, including their emails, comments, and posts.

```
function removeUser($mysqli, $username){
    //first find his UID
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);
    if(is_bool($first_row[0])){
        return 'removeUser: User with username '.$username.' does not
            exist.';
    }

    //now delete all:Emails, Event member, Group member, Content, Comment.
    Because there are foreign keys, I cannot replace certain values by
    bogus values
    //start with emails: easiest
    $mysqli->query("DELETE FROM Emails WHERE SourceUser='".$first_row[0]."'
        OR TargetUser='".$first_row[0]."'");
    //now go in post/post comment and find all content published by this
    user
    $result2 = $mysqli->query("SELECT CID FROM Post WHERE
        UID='".$first_row[0]."'");
    //do the same for post_comment
    $result3 = $mysqli->query("SELECT CoID FROM Post_Comment WHERE
        UID='".$first_row[0]."'");
    //find all comments on posts
    $result4 = $mysqli->query("SELECT Post_Comment.CoID FROM Post_Comment
        INNER JOIN Comment ON Comment.CoID=Post_Comment.CoID WHERE CID=
        ANY(SELECT CID FROM Post WHERE UID='".$first_row[0]."'");
    //remove all comments that are from a post of this user
    //then remove the post
    while($row = mysqli_fetch_row($result4)){
        $mysqli->query("DELETE FROM Post_Comment WHERE CoID='".$row[0]."'");
        $mysqli->query("DELETE FROM Comment WHERE CoID='".$row[0]."'");
    }

    while($row = mysqli_fetch_row($result3)){
```

```

    $mysqli->query("DELETE FROM Post_Comment WHERE CoID=".$row[0].";");
    $mysqli->query("DELETE FROM Comment WHERE CoID=".$row[0].";");
}

while($row = mysqli_fetch_row($result2)){
    $mysqli->query("DELETE FROM Post WHERE CID=".$row[0].";");
    $mysqli->query("DELETE FROM Content WHERE CID=".$row[0].";");
}

//now delete group/event membership
$mysqli->query("DELETE FROM Is_Member_Group WHERE
    UID=".$first_row[0].";");
$mysqli->query("DELETE FROM Is_Member_Event WHERE
    UID=".$first_row[0].";");

//finally, delete the user
$mysqli->query("DELETE FROM User_ WHERE UID=".$first_row[0].";");
return 'removeUser: '.$mysqli->error;
}

```

Remove a group and all its users.

```

function removeGroup($mysqli, $groupName){
    //must remove all content from group, all membership to group, and the
    group itself.
    $result = $mysqli->query("SELECT GroupID FROM Group_ WHERE
        GroupName='".$groupName."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'removeGroup: Could not find Group with GroupName
            \''.$groupName.'\'';
    }

    //find all posts belonging to the group
    $result2 = $mysqli->query("SELECT CID FROM Content WHERE
        GroupID=".$first_row[0].";");
    //now delete all

```

```

while($row=mysqli_fetch_row($result2)){
    $mysqli->query("DELETE FROM Post WHERE CID=".$row[0].";");
    $mysqli->query("DELETE FROM Content WHERE CID=".$row[0].";");
}

//then remove all membership
$mysqli->query("DELETE FROM Is_Member_Group WHERE
    GroupID=".$first_row[0].";");
//then remove group
$mysqli->query("DELETE FROM Group_ WHERE GroupID=".$first_row[0].";");
return 'removeGroup: '.$mysqli->error;
}

```

Remove an event and all its users.

```

function removeEvent($mysqli, $eventTitle){
    //find the eventID, then all the groups associated with it
    $result = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='".$eventTitle."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'removeEvent: Event with title '.$eventTitle.' was not
            found.';
    }

    $result_main = $mysqli->query("SELECT GroupID FROM Group_ WHERE
        MainEventID=".$first_row[0].";");

    while($row_main = mysqli_fetch_row($result_main)){
        //find all posts belonging to the group
        $result2 = $mysqli->query("SELECT CID FROM Content WHERE
            GroupID=".$row_main[0].";");
        //now delete all
        while($row=mysqli_fetch_row($result2)){
            $mysqli->query("DELETE FROM Post WHERE CID=".$row[0].";");
            $mysqli->query("DELETE FROM Content WHERE CID=".$row[0].";");
        }
    }
}

```

```

        //then remove all membership
        $mysqli->query("DELETE FROM Is_Member_Group WHERE
        GroupID=".$row_main[0].";");
        //then remove group
        $mysqli->query("DELETE FROM Group_ WHERE
        GroupID=".$row_main[0].";");
    }

    //now all groups are deleted. Only posts and comments from the event
    left.
    $result3 = $mysqli->query("SELECT CID FROM Content WHERE
        EventID=".$first_row[0].";");
    $result4 = $mysqli->query("SELECT CoID FROM Comment WHERE
        CID=ANY(SELECT CID FROM Content WHERE EventID=".$first_row[0].";");

    //remove all comments before
    while($row = mysqli_fetch_row($result4)){
        $mysqli->query("DELETE FROM Post_Comment WHERE CoID=".$row[0].";");
        $mysqli->query("DELETE FROM Comment WHERE CoID=".$row[0].";");
    }

    //then all content
    while($row = mysqli_fetch_row($result3)){
        $mysqli->query("DELETE FROM Post WHERE CID=".$row[0].";");
        $mysqli->query("DELETE FROM Content WHERE CID=".$row[0].";");
    }

    //then remove all membership to group
    $mysqli->query("DELETE FROM Is_Member_Event WHERE
        EventID=".$first_row[0].";");
    //then remove the event
    $mysqli->query("DELETE FROM Event_ WHERE EventID=".$first_row[0].";");
    return 'removeEvent: '.$mysqli->error;
}

```

NOTE: Queries have been modified to prepared statements to protect from SQL injection.

6. Specific test cases

The following are functions used in specific test cases.

Privacy is set to 0 when a group is private and 1 when it is public.

All functions are written in PHP, where:

- **\$mysqli** is used to connect to the database object
- **\$username** is the username of the user (string)
- **\$eventTitle** is the title of an event (must exist) (string)
- **\$groupName** is the name of a group (string)
- **\$UID** is the UID of a user (int)
- **\$eventTitle** is the title of an event (string)
- **\$CID** is the Content ID (int)
- **\$replyString** is the text (string) contained in a comment (can be empty) (string)
- **\$replyStringCID** is the text for the content being commented on (string)
- **\$replyImage** is the image contained in a comment (can be empty) (string)
- **\$sourceUsername** is the username of the sender of an email (string)
- **\$targetUsername** is the username of the receiver of an email (string)
- **\$date** is a date object of form 'YYYY-MM-DD'
- **\$title** is the name of an event (string)
- **\$expiryDate** is a date object of form 'YYYY-MM-DD'
- **\$eventType** is the type of an event (string)
- **\$usernameCreator** is the username of whoever created the event (string)
- **\$isProfitable** is a boolean determining if an event is profitable, 0 for no, 1 for yes (int)
- **\$numberEvents** is a positive integer that says how many events for a price (int)
- **\$storageGB** is the storage space the event takes in GB (int)
- **\$bandwidthGB** is the bandwidth an event takes in GB (int)
- **\$price** is the price for a package (double)
- **\$password** is the password of a user (string)
- **\$email** is the email address of a user (string)
- **\$name** is the full name of a user (first and last) (string)
- **\$dateofbirth** is the birth date of a user in format 'YYYY-MM-DD'
- **\$privilegelevel** 0 is standard privilege, 1 is for controller, 2 is for admin (int)

Confirming the creation of event. This happens when a user creates an event and needs confirmation from the administrator in order to get a page for the event. Only an admin can do this.

```
function confirmCreationEvent($mysqli, $eventTitle){
    //set status to 0 to put the event active.
    $mysqli->query("UPDATE Event_ SET Status=0 WHERE
        Title='".$eventTitle."'");
    return 'confirmCreationEvent: '.$mysqli->error;
}
```

Sets a user as a member of an event, from its pending state.

```
function setMemberToEvent($mysqli, $username, $eventTitle){
    //find userID
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'setMemberToEvent: User with username '.$username.' does not
            exist';
    }

    $result2 = $mysqli->query("SELECT EventID FROM Event_ WHERE
        Title='".$eventTitle."'");
    $first_row_2 = mysqli_fetch_row($result2);

    if(is_bool($first_row_2[0])){
        return 'setMemberToEvent: Event with event title '.$eventTitle.'
            does not exist.';
    }

    $mysqli->query("UPDATE Is_Member_Event SET requestStatus='member'
        WHERE UID='".$first_row[0]." AND EventID='".$first_row_2[0]."'");
    return 'setMemberToEvent: '.$mysqli->error;
}
```

Sets a user as a member of a group, from its pending state. Only available to group admins.

```
function setMemberToGroup($mysqli, $username, $groupName){
    //find userID
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'setMemberToGroup: User with username ' . $username . ' does not
            exist';
    }

    //find GroupID
    $result2 = $mysqli->query("SELECT GroupID FROM Group_ WHERE
        GroupName='".$groupName."'");
    $first_row_2 = mysqli_fetch_row($result2);

    if(is_bool($first_row_2[0])){
        return 'setMemberToGroup: Group with group name ' . $groupName . ' does
            not exist.';
    }

    $mysqli->query("UPDATE Is_Member_Group SET requestStatus='member'
        WHERE UID=".$first_row[0]. " AND GroupID=".$first_row_2[0].");
    return 'setMemberToGroup: ' . $mysqli->error;
}
```

Converts a date stamp to a timestamp, from 'YYYY-MM-DD' to a long.

```
function convertDateStampToTimeStamp($dateStamp){
    $a = strtotime($dateStamp, '%Y-%m-%d');
    return mktime(0, 0, 0, $a['tm_mon']+1, $a['tm_mday'],
        $a['tm_year']+1900);
}
```

Converts a timestamp to a date stamp, from a long to 'YYYY-MM-DD'.

```
function convertTimeStampToDateStamp($timestamp){
    return date('Y-m-d', $timestamp);
}
```

Verify the validity of an event (if it is archived).

```
function isEventArchived($mysqli, $eventTitle){
    $result = $mysqli->query("SELECT ExpiryDate FROM Event_ WHERE
        Title='".$eventTitle."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'isEventArchived: Event with title '.$eventTitle.' was not
            found.';
    }

    //current time
    $timestamp = time();

    //convert string to timestamp
    $event_timestamp = convertDateStampToTimeStamp($first_row[0]);

    return $event_timestamp < $timestamp? 1: 0;
}
```

Verify if an event should be deleted (archived for over 7 years).

```
function shouldEventBeDeleted($mysqli, $eventTitle){
    $result = $mysqli->query("SELECT ExpiryDate FROM Event_ WHERE
        Title='".$eventTitle."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'shouldEventBeDeleted: Event with title '.$eventTitle.' was
            not found.';
    }

    //increase by 7 the year on the expiry date
```

```

$year = strval(intval(substr($first_row[0],0,4))+7);
//then put this back in
$expiry_date = $year.substr($first_row[0],4);

//current time
$timestamp = time();

//convert string to timestamp
$sevent_timestamp = convertDateStampToTimeStamp($expiry_date);

return $sevent_timestamp < $timestamp? 1: 0;
}

```

Change the email of a user.

```

function updateUserEmail($mysqli, $username, $email){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUserEmail: User with username '.$username.' was not
        found.';
    }

    $mysqli->query("UPDATE User_ SET Email='".$email.'" WHERE
        Username='".$username."'");
    return 'updateUserEmail: '.$mysqli->error;
}

```

Change the password of a user.

```

function updateUserPassword($mysqli, $username, $password, $new_password){
    $result= $mysqli->query("SELECT UID, Password FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUserPassword: User with username '.$username.' was not
        found.';
    }
}

```

```

}

if(strcmp($password, $first_row[1])!=0){
    return 'updateUserPassword: Wrong credentials!';
}

$mysqli->query("UPDATE User_ SET Password='".$new_password.'" WHERE
    UID='".$first_row[0]."';");
return 'updateUserPassword: '.$mysqli->error;
}

```

Change the name of a user.

```

function updateUser_Name($mysqli, $username, $name){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUser_Name: User with username '.$username.' was not
            found.';
    }

    $mysqli->query("UPDATE User_ SET Name='".$name.'" WHERE
        Username='".$username."'");
    return 'updateUser_Name: '.$mysqli->error;
}

```

Change the bankName of a user.

```

function updateUserBankName($mysqli, $username, $bank_name){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUserBankName: User with username '.$username.' was not
            found.';
    }

    $mysqli->query("UPDATE User_ SET BankName='".$bank_name.'" WHERE
        Username='".$username."'");
}

```

```

    return 'updateUserBankName: '.$mysqli->error;
}

```

Change the creditCardNumber of a user.

```

function updateUserCreditCardNumber($mysqli, $username,
$credit_card_number){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUserCreditCardNumber: User with username '.$username.'
        was not found.';
    }
    $mysqli->query("UPDATE User_ SET
        CreditCardNumber='".$credit_card_number.'" WHERE
        Username='".$username."'");
    return 'updateUserCreditCardNumber: '.$mysqli->error;
}

```

Change the AccountNumber of a user.

```

function updateUserAccountNumber($mysqli, $username, $account_number){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUserAccountNumber: User with username '.$username.'
was not found.';
    }
    $mysqli->query("UPDATE User_ SET AccountNumber='".$account_number.'"
WHERE Username='".$username."'");
    return 'updateUserAccountNumber: '.$mysqli->error;
}

```

Change the phoneNumber of a user.

```
function updateUserPhoneNumber($mysqli, $username, $phone_number){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUserPhoneNumber: User with username '.$username.' was
            not found.';
    }
    $mysqli->query("UPDATE User_ SET PhoneNumber='".$phone_number."' WHERE
        Username='".$username."'");
    return 'updateUserPhoneNumber: '.$mysqli->error;
}
```

Change the address of a user.

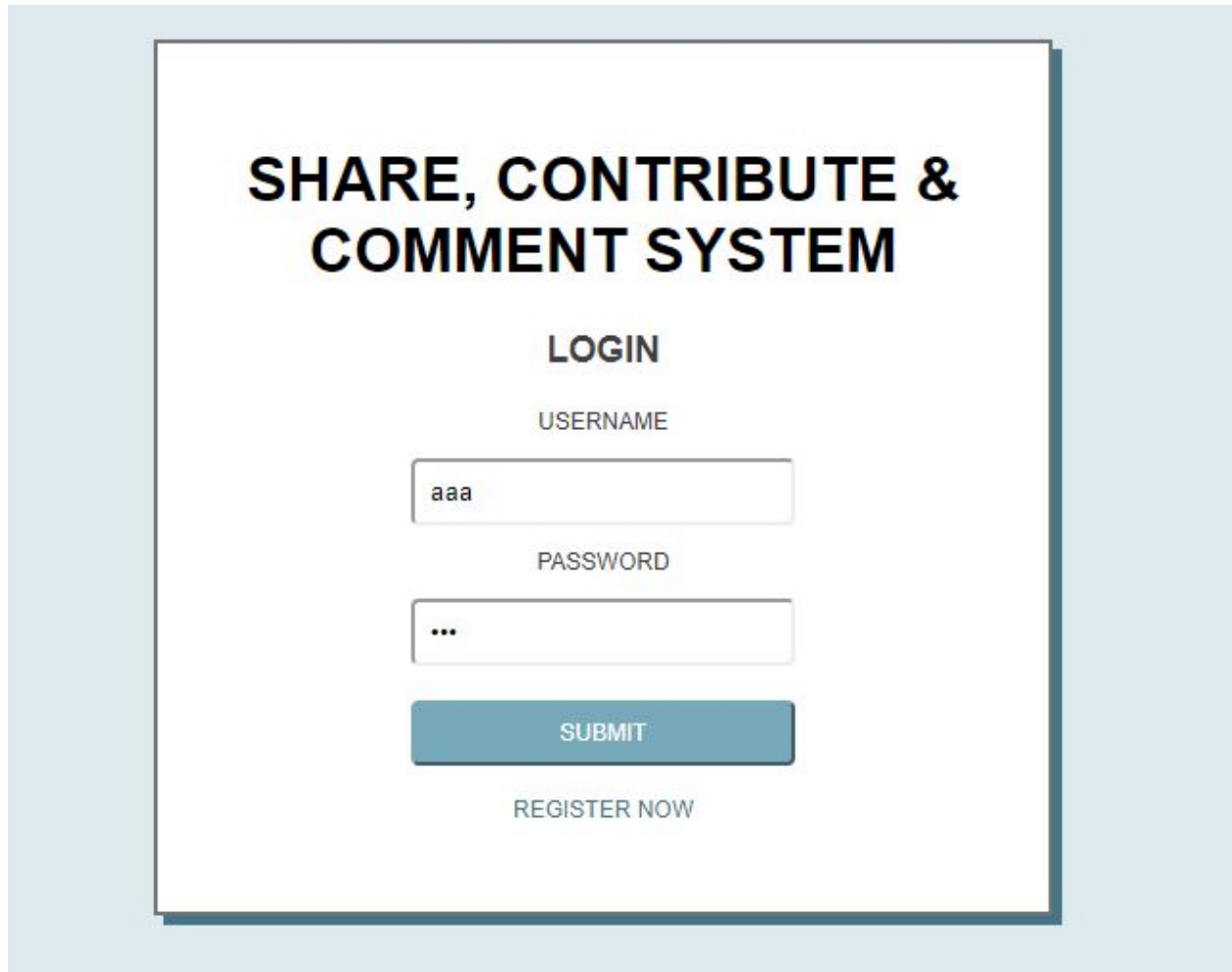
```
function updateUserAddress($mysqli, $username, $address){
    $result = $mysqli->query("SELECT UID FROM User_ WHERE
        Username='".$username."'");
    $first_row = mysqli_fetch_row($result);

    if(is_bool($first_row[0])){
        return 'updateUserAddress: User with username '.$username.' was not
            found.';
    }
    $mysqli->query("UPDATE User_ SET Address='".$address."' WHERE
        Username='".$username."'");
    return 'updateUserAddress: '.$mysqli->error;
}
```

NOTE: Queries have been modified to prepared statements to protect from SQL injection.

7.0 Sample session & UI guide

7.1 Home screen

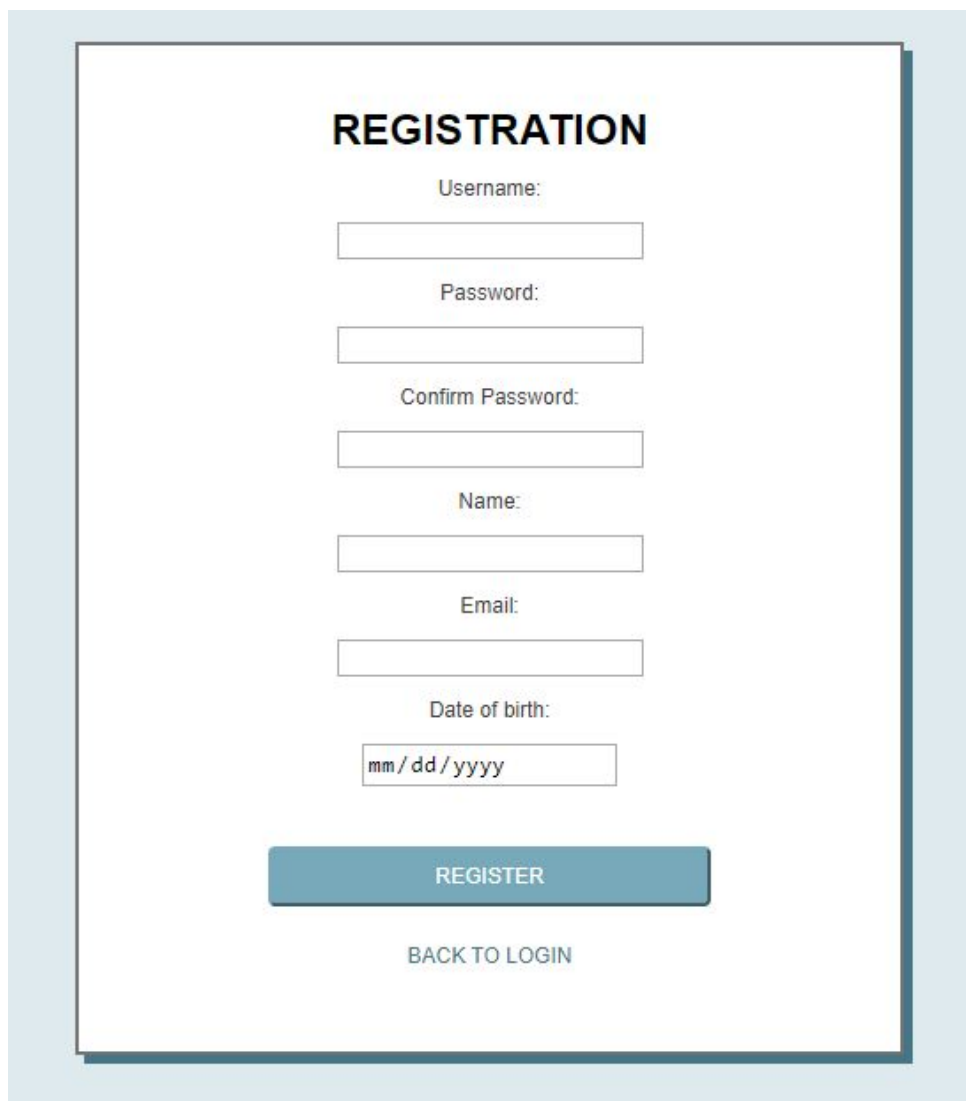


The image shows a mobile application home screen with a light blue background. A white rectangular card is centered on the screen. At the top of the card, the text 'SHARE, CONTRIBUTE & COMMENT SYSTEM' is displayed in a large, bold, black sans-serif font. Below this, the word 'LOGIN' is centered in a smaller, bold, black font. Underneath 'LOGIN', the label 'USERNAME' is centered in a light gray font. Below the label is a white text input field with a thin gray border, containing the text 'aaa'. Below the input field, the label 'PASSWORD' is centered in a light gray font. Below the label is another white text input field with a thin gray border, containing three dots '...'. Below the password input field is a solid blue rectangular button with the word 'SUBMIT' centered in white, uppercase, sans-serif font. At the bottom of the card, the text 'REGISTER NOW' is centered in a light gray font.

Upon opening the application, the user is greeted with the above screen. They can either login with their existing credentials or register for a new account using the 'Register now' link. For registering see **7.2**.

7.2 Registering

Here users can register for a new account by filling out the information on the form and submitting it. They can then return to the home screen and log in using the credentials they have just created. User is required to fill out all fields otherwise the application will not allow user to register. Note that when a user writes their password, the password is encrypted using bcrypt into the database.

A registration form titled "REGISTRATION" is displayed within a light blue container. The form is centered and contains several input fields and buttons. The fields are labeled "Username:", "Password:", "Confirm Password:", "Name:", "Email:", and "Date of birth:". The "Date of birth:" field has a placeholder "mm/dd/yyyy". Below the fields is a large blue "REGISTER" button, and at the bottom is a smaller "BACK TO LOGIN" link.

REGISTRATION

Username:

Password:

Confirm Password:

Name:

Email:

Date of birth:

REGISTER

[BACK TO LOGIN](#)

7.3 Home page

Once the user has successfully logged in, they are directed to the above screen, the home page. On this page they can search for an event (7.6) or user, find links to edit their user information (7.4) and group/event details (7.5), send an email (7.9) see their own user data, their upcoming events, their groups, and log out. You can select rows of the tables emails/events/groups to show their content. On hover, they will change color.

Welcome aaa

USER DATA

| | |
|----------------|------------|
| SCC User ID: | 1 |
| Name: | abl arr |
| Username: | aaa |
| Email: | a@a.com |
| Date Of Birth: | 1777-04-05 |

EMAILS RECEIVED

| Source User | Email Title | Content |
|-------------|-------------|----------|
| aaa | aaa | asdfasdf |
| aaa | ye | yee |

EMAILS SENT

| Target User | Email Title | Content |
|-------------|-------------|----------|
| aaa | aaa | asdfasdf |
| aaa | ye | yee |

UPCOMING EVENTS

| Event ID | Title | Latest Post | Latest Post Timestamp |
|----------|------------|--------------------|-----------------------|
| 1 | Some_Event | Hail is the worst! | 2019-12-05 06:19 |
| 3 | famjam | | |
| 6 | he!0 | | |

YOUR GROUPS

| Group ID | Title | Main Event ID | Latest Post | Latest Post Timestamp |
|----------|-------------|---------------|-------------|-----------------------|
| 1 | Org | 1 | heyo | 2019-12-09 06:49 |
| 2 | Groupaaaaaa | 1 | oy ayoye | 2019-12-09 06:14 |

7.4 Editing user information

Edit user information

Current email: a@a.com
Change email

Password: hidden for your privacy
Change password

Current name: abl arr
Change name

Bank name: hidden for your privacy
Change bank name

Credit card number: hidden for your privacy
Change credit card number

Account number: hidden for your privacy
Change account number

Current address: 123 sesame street
Change address

Current phone number: 51412345678
Change phone number

Delete user account

If the user clicks on 'Edit User Information' from the home screen, they are brought to the above screen. Here the user can edit all of their personal information. Sensitive data such as their current password, bank name and banking/card information is hidden for their privacy. The user can also delete their account, or simply return to the homepage using the button at the bottom.

7.5 Editing group/event details

The screenshot shows a web interface titled "Edit group/event details". It is divided into three main sections: "EVENTS", "GROUPS", and "ADD AN EVENT".

- EVENTS**: Displays "Event Some_Event:" followed by a blue button labeled "DELETE EVENT".
- GROUPS**: Displays "Group Org:" followed by a blue button labeled "LEAVE GROUP".
- ADD AN EVENT**: Contains a form with three fields: "Event Name" (text input with placeholder "Event Name..."), "Event Type" (dropdown menu with "community" selected), and "Event Template" (text input with placeholder "Template Selection (1 or 2)"). Below these fields are two blue buttons: "ADD AN EVENT" and "RETURN TO HOMEPAGE".

If the user clicks on 'Edit Group/Event Details', they are brought to the above screen. Here they can see which events they have created and/or are managing and have the option to delete them. They can also see the

groups they are a part of and decide to leave them. Underneath this is the option to add an event. To do so they must have entered their complete banking information in 7.4. They must then choose a name, event type and event template. They can return to the homepage as well. Clicking on 'Add an Event' will show a payment screen to pay for the event. The payment system may have some issues, but a safeguard measure was made to add the event even though the user has not paid in case the API keys do not function anymore.

7.5.1 Adding an event that is not non-profit

Disclaimer: By applying this rate, you will get 25GB of storage and 25GB of bandwidth. An extra payment of 3\$ and 3\$ applies every time bandwidth and storage goes over. The price per year for the event being active is 40\$.

The charge for the rate selected and this type of event (community) is 40\$. You can click here to finalize the event.

Please enter the necessary information.

Charge \$80 CAD with 2Checkout.

NAME

EMAIL

CARD NUMBER

EXPIRY DATE MM YY

CVV

SUBMIT PAYMENT

If a user adds an event from the above screen that is not a non-profit event, they will be requested to pay a certain rate. They must fill in the above form and click on submit payment.

7.5.2 Successful payment

M2EyY2YyNGUtYjQ3OC00NDA4LTg4NWUtNGY5MjZkZGRmZTcx

Thanks for your Order!

The transaction was successful. Order details are given below:

Order ID: SKA92712382139

Order Number: 8342304234

Transaction ID: 23402u304923

Order Total: 20.00 CAD

[Back to Home Page](#)

Once the user has successfully paid for their Event, they are brought to their order success screen, displaying important information related to the transaction and order. They can then return to the homepage.

7.6.1 Event Page Not Found

Oops! It looks like this event does not exist yet...

If a user searches for an event that does not exist, they are brought to this page. They must return to the homepage afterwards. This screen might also be shown with another text message mentioning something about the event not being approved. This is because the event has been paid for by the user, but not yet approved by the administrator to be shown.

7.6.2 Event Page (participant)

Archived

Some_Event

| Event Manager Details | | |
|-----------------------|---------|--------------|
| Name | Email | Phone Number |
| abl arr | a@a.com | 51412345678 |

GROUPS

| Group Name | Group Admin | Action |
|------------|-------------|--|
| Org | bbb | <input type="button" value="SEE GROUP CONTENT"/> |

| Username | Content | Timestamp |
|----------|--|------------------|
| aaa | Hail is the worst! | 2019-12-05 06:19 |
| aaa | So how about we set this up for the 5th of December? | 2019-12-05 06:19 |
| bbb | Sounds good to me! | 2019-12-05 06:19 |
| bbb | But it would need to be in the evening | 2019-12-05 06:19 |
| aaa | Howdy yall! Zis my new event | 2019-12-05 06:19 |

If a user searches for an existing event, they are brought to the event page. Here they have the option to search for another event or user, return to the homepage or view the Event details. If the event is archived it will say so at the top above the title. This is known by looking at the expiring date of the event. They can view the details of the event manager, see a list of all members of the event (7.6.2.1), see the groups and group content, and see all content posted in the group. They can also click on “See Group Content” in a group to see a group conversation (7.6.2.2) Because the event is being archived, the user cannot add content to the event (in terms of comments, which is the username preceded by _____, or in terms of posts, which is without the _____. Only the event manager is allowed to do posts).

7.6.2.1 Members of an event

The screenshot shows a web interface for an event. At the top, there is a search bar with the placeholder text "Enter event or user..." and two buttons: "SEARCH EVENT" and "SEARCH USER". Below these is a "RETURN TO HOMEPAGE" button. The event title is "Some Event", and it is marked as "Archived". A modal window titled "MEMBERS OF EVENT" is open, displaying a list of members:

| Username | Role | Actions |
|----------|--------|---------------------------|
| aaa | admin | SEND MESSAGE HOME PAGE |
| bbb | member | SEND MESSAGE HOME PAGE |
| ccc | member | SEND MESSAGE HOME PAGE |

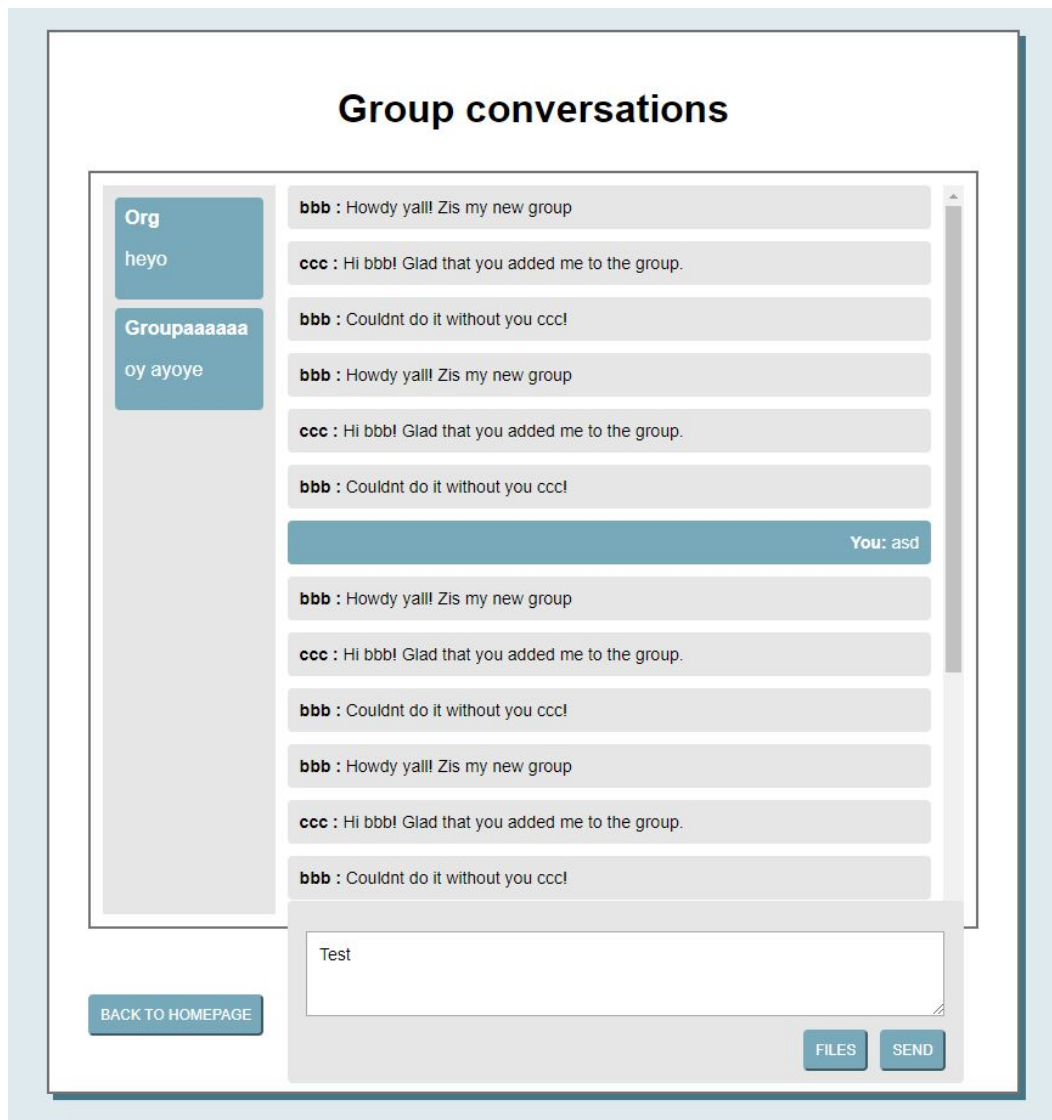
Below the modal, there is a "CLOSE" button. At the bottom of the page, there is a table showing the event's content:

| Username | Content | Timestamp |
|----------|--|------------------|
| aaa | Hail is the worst! | 2019-12-05 06:19 |
| aaa | So how about we set this up for the 5th of December? | 2019-12-05 06:19 |
| bbb | Sounds good to me! | 2019-12-05 06:19 |
| bbb | But it would need to be in the evening | 2019-12-05 06:19 |
| aaa | Howdy yall! Zis my new event | 2019-12-05 06:19 |

Here users can view all members of an event and opt to send them a message if so desired. By sending a message to any user, that message

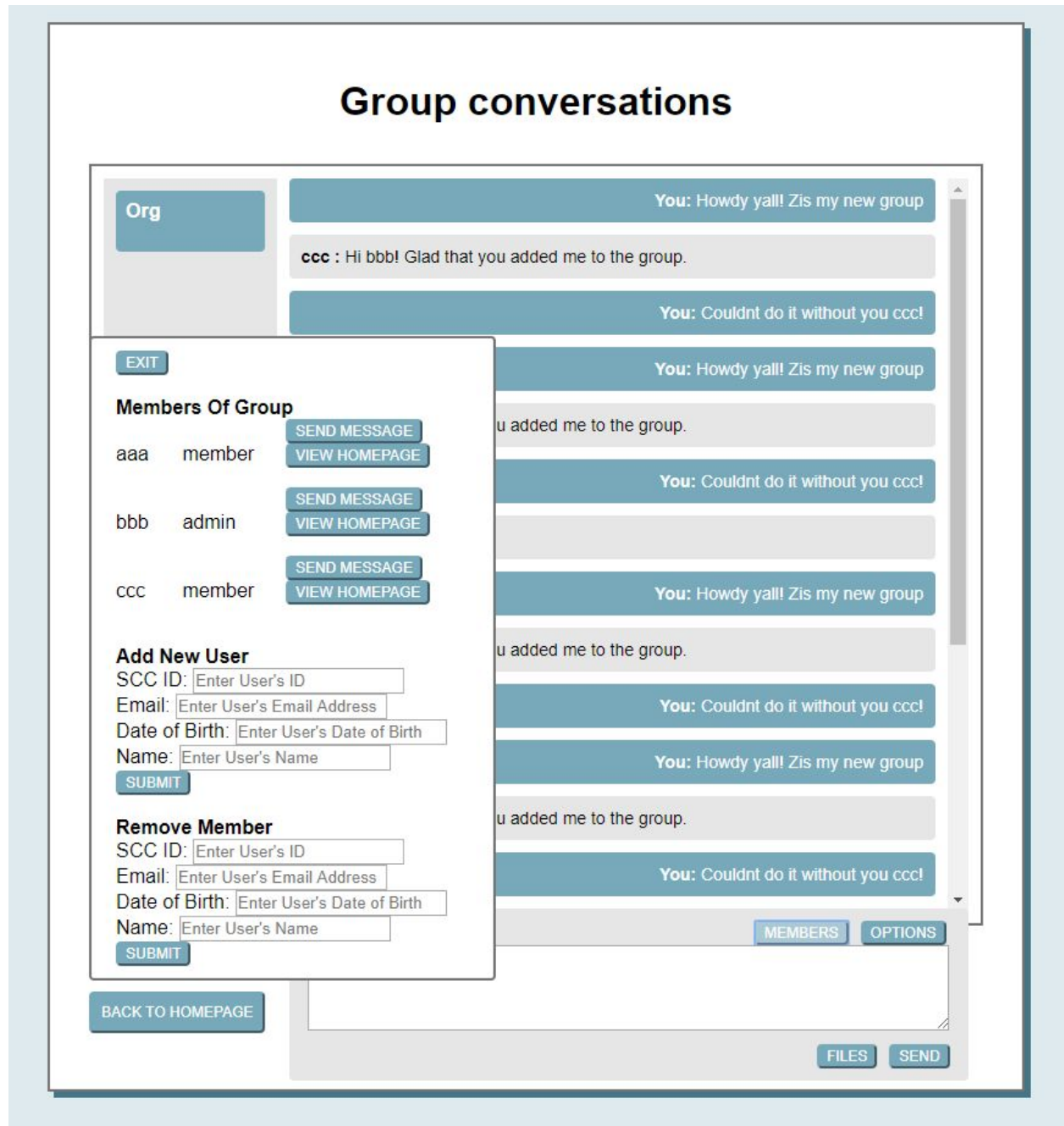
creates a new private group with name 'Group'+user1+user2. The user that sends the message first becomes the administrator. The other becomes a member. The 'Home page' button redirects to another user's home page. It can be yours as well.

7.6.2.2 Group conversations



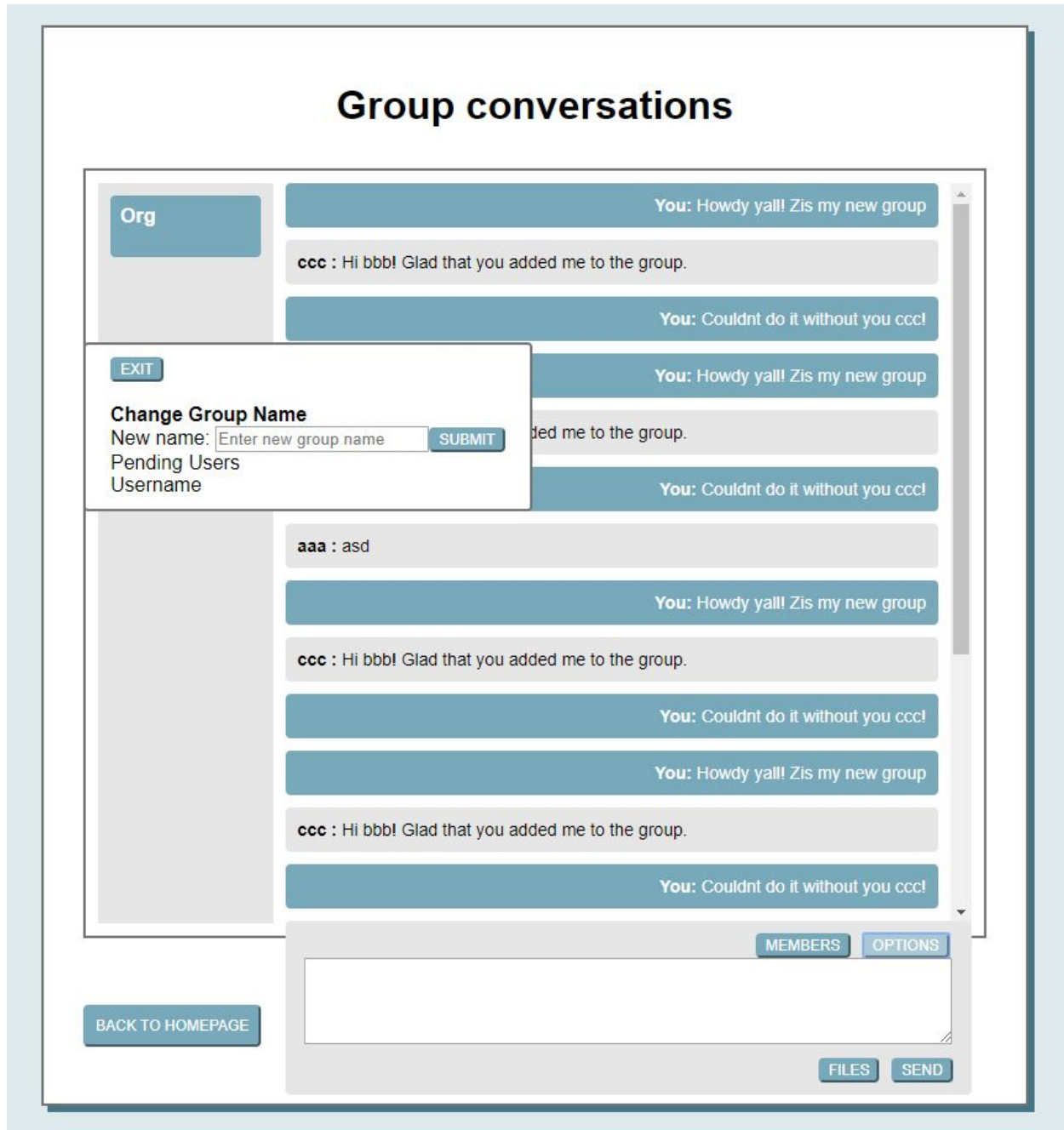
Here users can select a group on the left hand side and have a discussion via text messages with the group members. They simply type what they want to share in the text box and click send. They can also attach files to their messages. The user's own messages are sent in blue, while the received messages are in grey. The sender's name is in bold. By being the group manager, two new buttons appear below: 1 for members, which allows you to add new members, and see the current members of the group. The other is options, which allows you to change the name of the group and accept pending users which want to join your group.

If a participant is the group admin, they have access to two additional buttons, “Members” and “Options”, as seen above the textbox in the screenshot below.



7.6.2.3 Members window

User clicks the “Members button”. Here the group admin can view and message members, as well as add and remove them.



7.6.2.4 Group Options

User clicks the “Options” button. Here the user can rename the group, as well as see pending users to the group and approve / remove them.

7.6.3 Event page (manager)

Hyperparty

| Event Manager Details | | |
|-----------------------|-------------------------|--------------|
| Name | Email | Phone Number |
| Alexandre | alexandre1@concordia.ca | 4504496784 |

GROUPS

| Group Name | Group Admin | Action |
|------------|-------------|--|
| Borui Town | alexandre | <input type="button" value="SEE GROUP CONTENT"/> |

| Username | Content | Timestamp |
|--|---------------------------------------|-----------|
| Reply: <input type="text" value="Your Reply"/> | <input type="button" value="Submit"/> | |
| <input type="text" value="No comments allowed"/> | | |
| No content to show | | |

The event manager can see and do everything a participant can, as well as the option to see pending members wishing to join the event (7.6.3.1) and request for a new member to join the event (7.6.3.2).

7.6.3.1 Viewing pending members

Archived

Some Event
 PENDING USERS

| | | |
|---------|---------|-------------|
| abl arr | a@a.com | 51412345678 |
|---------|---------|-------------|

GROUPS

| Group Name | Group Admin | Action |
|------------|-------------|--|
| Org | bbb | <input type="button" value="SEE GROUP CONTENT"/> |

| Username | Content | Timestamp |
|----------|--|------------------|
| aaa | Hail is the worst! | 2019-12-05 06:19 |
| aaa | So how about we set this up for the 5th of December? | 2019-12-05 06:19 |
| bbb | Sounds good to me! | 2019-12-05 06:19 |
| bbb | But it would need to be in the evening | 2019-12-05 06:19 |
| aaa | Howdy yall! Zis my new event | 2019-12-05 06:19 |

Here an event manager can view pending users awaiting acceptance to the event. By clicking on 'Accept' showing next to the user's username, the user will join the event.

7.6.3.2 Requesting to add a new member

Enter event or user... SEARCH EVENT SEARCH USER

RETURN TO HOMEPAGE

Archived

Some Event

ADD NEW USER

SCC ID: Enter User's ID

Email: Enter User's Email Address

Date of Birth: Enter User's Date of Birth

Name: Enter User's Name

SUBMIT

CANCEL

| Group Name | Group Admin | Action |
|------------|-------------|-------------------|
| Org | bbb | SEE GROUP CONTENT |

| Username | Content | Timestamp |
|----------|--|------------------|
| aaa | Hail is the worst! | 2019-12-05 06:19 |
| aaa | So how about we set this up for the 5th of December? | 2019-12-05 06:19 |
| bbb | Sounds good to me! | 2019-12-05 06:19 |
| bbb | But it would need to be in the evening | 2019-12-05 06:19 |
| aaa | Howdy yall! Zis my new event | 2019-12-05 06:19 |

Here an event manager can add a new member to an event by filling in their information in this window and clicking submit. All the information must be accurate, otherwise adding a new user will fail. Note that by adding a new user, a one time code is sent via email to the user's mailbox. That user receiving the request must copy that one time code, go to the event's page, and paste the one time code into the corresponding box. He will then get access to the event once he validates his one time code.

7.7 Controller home page

Current Rates

| Rates ID | Number Events | Event Type | Storage GB | Bandwidth GB | Price | Bandwidth Overflow Fee | Storage Overflow Fee |
|----------|---------------|------------|------------|--------------|-------|------------------------|----------------------|
|----------|---------------|------------|------------|--------------|-------|------------------------|----------------------|

ADD RATES

Number of Events User Created:

Event Type:

Storage in GB Offered:

Bandwidth in GB Offered:

Price:

Price per GB for Overflowing Bandwidth:

Price per GB for Overflowing Storage:

DELETE RATES

RID of the Rate:

When a controller logs into the application, they can access this page where they can add and delete rates for Events. This is only accessible by Controllers. The number of events parameter determines how many events must a user create to be eligible for a certain rate. The event type determines which type of event must the user be creating to be eligible to a certain rate. The storage in GB is the storage limit the event can take. The bandwidth in GB is the bandwidth limit the event can take. The bandwidth overflow and storage overflow fees are for when the user surpasses the bandwidth or storage suggested by this rate.

7.8 Admin home page

There are many options for the admin on their homepage. This is only accessible by Admins.

SYSTEM USERS

| UID | Username |
|-----|----------|
| 9 | 123 |
| 1 | aaa |
| 2 | bbb |
| 5 | ccc |
| 4 | ppp |
| 3 | rrr |

SYSTEM GROUPS

| GroupID | Group Name |
|---------|-------------|
| 2 | Groupaaaaaa |
| 1 | Org |

SYSTEM EVENTS

| EventID | Event Title | Is Pending |
|---------|-------------|------------|
| 1 | Some_Event | 0 |
| 3 | famjam | 1 |

First they are displayed the system's users, groups and events. They can use the above information to complete any of the following actions.

REMOVE USER FROM GROUP

Username:

GroupID:

REMOVE

REMOVE USER FROM EVENT

Username:

EventID:

REMOVE

They can remove users from groups and events using their username and the group or event ID.

The image displays three distinct forms stacked vertically, each with a title and input fields. The first form, 'APPROVE EVENT', has an 'Event ID' input field and an 'APPROVE' button. The second form, 'SEND CONTENT ON EVENT', has 'Event Title' and 'Content' input fields and a 'SEND' button. The third form, 'SEND CONTENT ON GROUP', has 'Group ID' and 'Content' input fields and a 'SEND' button. All forms are flanked by light blue vertical bars.

APPROVE EVENT

Event ID:

APPROVE

SEND CONTENT ON EVENT

Event Title:

Content:

SEND

SEND CONTENT ON GROUP

Group ID:

Content:

SEND

They can approve events using the Event ID, as well as send content on an Event and/or a Group.

The image shows a single form titled 'SET OTHER USER AS MANAGER'. It contains two input fields: 'Event ID' and 'User ID'. Below the fields is a 'SET' button. The form is flanked by light blue vertical bars.

SET OTHER USER AS MANAGER

Event ID:

User ID:

SET

They can also set another user as a manager using the Event ID and the targeted users user ID.

The screenshot shows two forms within a light blue bordered container. The first form, titled 'REMOVE A GROUP', has a label 'Group ID:' followed by a text input field containing 'Group ID'. Below this is a red button labeled 'REMOVE'. The second form, titled 'REMOVE AN EVENT', has a label 'Event ID' followed by a text input field containing 'Event ID'. Below this is another red button labeled 'REMOVE'. At the bottom of the container is a blue button labeled 'RETURN TO HOME PAGE'.

Lastly, the admin can remove a Group or Event using the Group or Event ID. They can also return to the home page.

7.9 Sending an email

The screenshot shows a form titled 'SEND AN EMAIL' within a light blue bordered container. The form has four fields: 'From:' with the value 'aaa', 'To:' with the placeholder 'User destination...', 'Title:' with the placeholder 'Title of email...', and 'Content' with the placeholder 'Content email...'. Below these fields are two blue buttons: 'SEND' and 'RETURN TO HOME PAGE'.

Here a user can send an email to other users on the system, or return to the homepage. This page can be accessed from the home page.

8.0 Some Possible Scenarios

Add a User to an Event (from Event Manager)

- Log in as the event manager of an event.
- Go on that event page and select the 'Request a User to Join' button
- Fill in the input fields with the correct information from a user
- Then click on Submit.
- Log in as the user you want to add to the event.
- Check your mail; the one time code will be in your mail.
- Once you found it, go on the event page and enter your one time code in the corresponding field and click on 'Submit'.

Add a User to a Group (from Group Manager) - The user being added must be member of the event the group belongs in

- Log in as the group manager of the group.
- Go on the event page for which the group manager is the admin of a group.
- Select 'See Group Content' of the group he/she is administrator of.
- From the groupConversations.php file, select the group the user is administrator of.
- Click on the button 'Members'.
- Fill in the form under 'Add New User'.
- Click on 'Submit'.
- Log in as the user that receives the invitation.
- Go to their home page and check the emails; there should be one with a one time code.
- Then, go on the event page.
- Find the group you received a one time code for and fill in the blank next to the group name in the list of groups. Confirm your one time code.

Add a User to an Event (from User)

- Log in as a user not registered to the event.
- Go to the event page.

- Click on the button 'Join Event'
- Log in as the event manager.
- Go to the event page.
- Click on the button 'See Pending Requests'.
- Accept the user that just clicked on 'Join Event' by clicking right next to the user's username in the list the button 'Accept as Member'.

Add a User to a Group (from User) - The user must be a member of the event the group belongs in

- Log in as a user not registered to the group.
- Go to the event page.
- Click on the button 'Join Group'.
- Log in as group manager.
- Go to the event page.
- Click on 'See Group Content' on the group the user is admin of.
- On the groupConversations.php page, select the group for which you are an admin of.
- Click on 'Options'.
- Select the pending user from the list and click on 'Accept as Member'.

Add an Event

- Log in as a user.
- Go on your home page and select the link 'Edit event/group details'
- Go in the 'Add an Event' section and fill in the information required.
- Once you click on 'Submit', you will be directed to a payment page. Enter your information.
- Once this is done, you have a payment's receipt, and the information is sent to the administrator. (Your event cannot be displayed yet)
- Log in as the administrator.
- Go on your home page and select the link 'Manage System'.
- Put the event id where it says 'Approve Event' and click on 'Approve'.
- If you search for the event using the search bar, the event will now appear.

Add a Group

- Log in as a user.
- Go on your home page and search for an event you are member of using the search bar.
- Click on 'See Members': You will now get a list of users.
- Select the button 'Send Message' next to a user's name. (A new private group is created)
- (to make this group public to the event...) Log in as the event manager.
- Go to the event page using the search bar.
- Click on 'View Hidden Groups'.
- Click on 'Put Public' to make the group public on the page.

The rest of the scenarios are more straightforward.

9.0 Installation Guide and testing

1. Install and open XAMPP and start both Apache and MySQL
2. Place all the project files in C:\xampp\htdocs
3. Open any html files to test for functionality

9.1 connecting to database via COMMAND PROMPT

1. Type `ssh ENCSUSERNAME@login.encs.concordia.ca` (replace ENCSUSERNAME with your ENCS User name)
2. Enter your ENCS password (same as computer lab)
3. Type `mysql -h urc353.encs.concordia.ca -u urc353_2 -p urc353_2`
4. Enter the password AqtjPG

9.2 How to setup your environment

1. Use the script 'createTablesRevised.php' to create all the different tables required by the project.
2. Then use the script 'insertIntoTables.php' to populate the tables; although in this case, the administrator username and password will be 'systemAdmin' and 'systemAdmin', the controller username and password will be 'alexandre' and 'alexandre', and a regular user username and password would be 'username1' and 'password1'
3. To run the project, launch your XAMPP("for testing purposes") server, and go over the files that are found directly under the html/ folder. The requests/ folder is used for requests to the backend, and therefore does not compose the GUI. The other files are css and js which complement the view.

10. README

Group ID

29

Group Name

COMP 353_group_29

Group Members

Mair ELBAZ, 40004558, m_elba@encs.concordia.ca

Daniel VIGNY-PAU, 40034769, d_vignyp@encs.concordia.ca

Francois DAVID, 40046319, f_avid@encs.concordia.ca

Alexandre THERRIEN, 40057134, a_errie@encs.concordia.ca

Charles-Antoine GUITÉ, 40063098, c_guite@encs.concordia.ca

group account

urc353_2

password

AqtjPG

Project URL

<https://urc353.encs.concordia.ca/index.html>

<https://github.com/alt440/DatabasesPHPMysql/tree/master/FinalProject>

UserID and Password

Admin: username: *rrr* & password: *rrr*

Controller: username: *controller* & password: *controller*

Regular users: username: *aaa* & password: *aaa* (blank user, has just been created), username: username1 & password: password1, username: username2 & password: password2

11. Contributions

Mair Elbaz

- E/R Diagram
- Test case involving a Participant to join Event
- Requests/Add/Remove page
- CSS

Daniel Vigny-Pau

- E/R Diagram
- Test cases about the event manager assign permissions to a post, sub-reply to comment and create groups.
- Documentation
- CSS and page structure

Francois David

- E/R Diagram
- Test case where the participant creates an event, status pending until approval of the system admin.
- Login, Registration and Home page.
- CSV to load database instances.
- Payment system page and processing with 2checkout.

Alexandre Therrien

- E/R Diagram + Final touches
- PHP requests files, dynamic content
- All of the database layer: Functions which makes calls to the database.
- Documentation

Charles-Antoine Guite

- E/R Diagram
- Test case where event manager sends request to participants.
- Group Conversation Page