Assignment 3
EECS 3421M

Ali Tashfiq Wazed
ID: 214677405
tashfiq@my.yorku.ca
EECS username: **tashfiq**

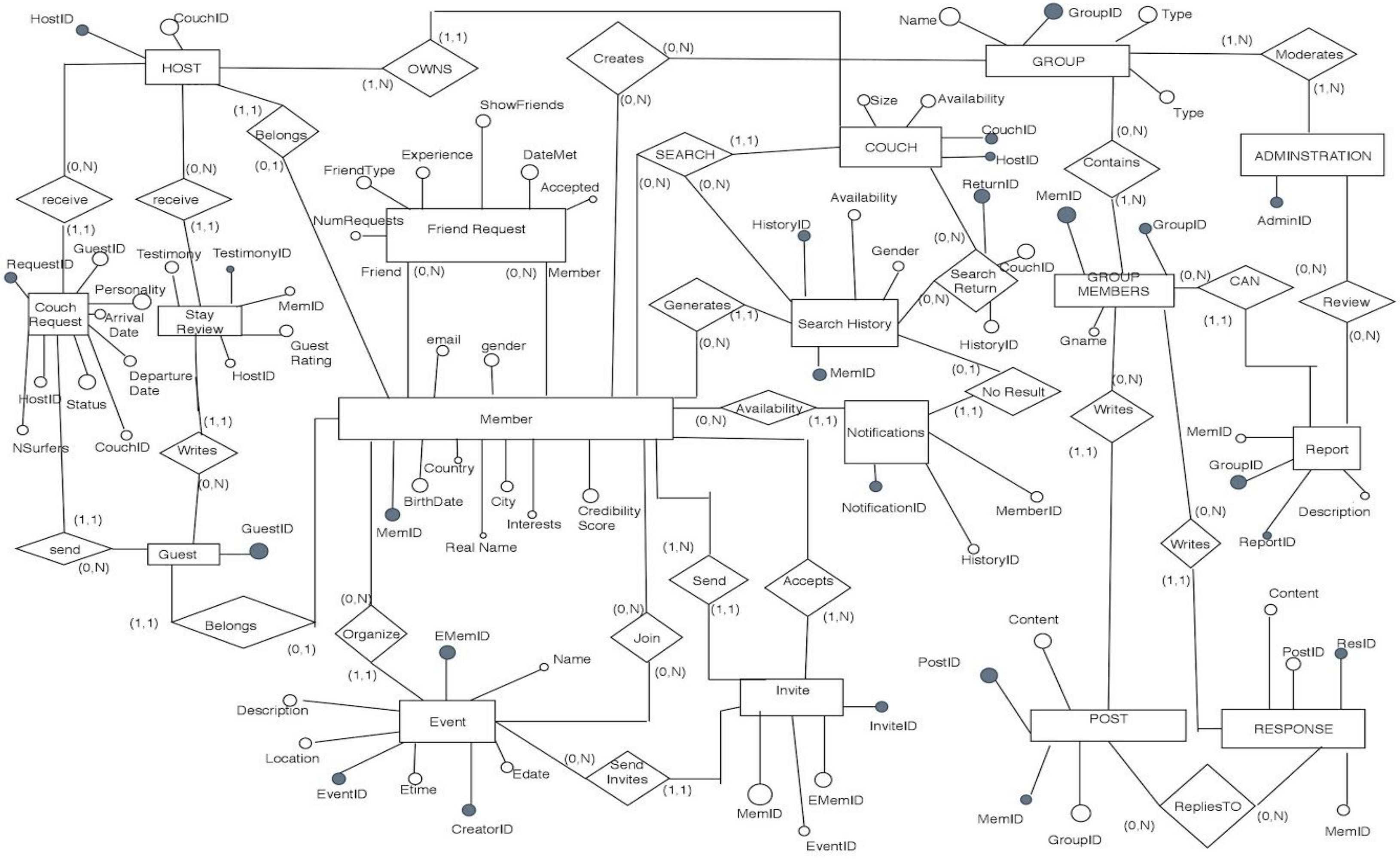
Shifat Akter
ID: 215162183
shifata@my.yorku.ca
EECS username: **shifata**

Grace Days Used: 02

Part A, Assumptions:

- No member can be friends (send friendship request) to administrators.
- Posts and Responses are written by Members
- A Guest can only rate a host ,write testimonies and send couch requests once.
- Every member is already signed up and in the system
- When a member joins an event, we are assuming they are attending the event and can invite other members
- Availability means the timeline on which the couch is available
- A report can be reviewed and moderated by many different administrators
- We have a counter “no.of requests” that allows the member to send a certain number of requests per day.
- StayReview incorporates both testimony and guest rating
- Interests incorporates pets, music and movie choices
- Size in couch refers to number of rooms in the house of the host
- In the notifications, we will notify members depending on their previous search criteria
- Members cannot send request to themselves.
- Member can be a host and a guest, it depends on the actions they're trying to make
- You can write up to 500 characters for your interest
- In ‘Invite’ , EMemID are members who are attending the event. While MemID are members who are being invited by EMemID.

Part B, An Entity Relationship Diagram (ERD) --- next page


HOST
HostID
CouchID
OWNS
(1,1)
(1,N)
Belongs
(1,1)
(0,1)
receive
(0,N)
(1,1)
receive
(0,N)
(1,1)
Couch Request
RequestID
GuestID
Personality
Arrival Date
Departure Date
HostID
Status
NSurfers
CouchID
Stay Review
Testimony
TestimonyID
MemID
Guest Rating
HostID
Writes
(1,1)
(0,N)
send
(1,1)
(0,N)
Guest
GuestID
Belongs
(1,1)
(0,1)
Creates
(0,N)
(0,N)
Friend Request
ShowFriends
Experience
DateMet
FriendType
Accepted
NumRequests
Friend
(0,N)
(0,N)
Member
Member
email
gender
Country
BirthDate
City
Interests
Credibility Score
MemID
Real Name
SEARCH
(1,1)
(0,N)
(0,N)
Generates
(1,1)
(0,N)
Search History
HistoryID
Availability
Gender
MemID
Search Return
(0,N)
(0,N)
ReturnID
CouchID
HistoryID
No Result
(0,1)
(1,1)
Availability
(0,N)
(1,1)
Notifications
NotificationID
MemberID
HistoryID
COUCH
Size
Availability
CouchID
HostID
GROUP
Name
GroupID
Type
Type
Moderates
(1,N)
(1,N)
ADMINSTRATION
AdminID
Contains
(0,N)
(1,N)
GROUP MEMBERS
MemID
GroupID
Gname
CAN
(0,N)
(1,1)
Review
(0,N)
(0,N)
Report
MemID
GroupID
Description
ReportID
Writes
(0,N)
(1,1)
Writes
(0,N)
(1,1)
POST
PostID
Content
MemID
GroupID
RepliesTO
(0,N)
(0,N)
RESPONSE
Content
PostID
ResID
MemID
Organize
(0,N)
(1,1)
Event
EMemID
Name
Description
Location
EventID
Etime
Edate
CreatorID
Join
(0,N)
(0,N)
Send
(1,N)
(1,1)
Accepts
(1,N)
Send Invites
(0,N)
(1,1)
Invite
InviteID
MemID
EMemID
EventID

Part C Relational Schema

Member (MemID, BirthDate, RealName, Country, City, Interests, CredibilityScore, email, gender)
Invite (InviteID, **EMemID**, **MemID**, EventID)
Event(EventID, CreatorID, EMemID, description, Edate, location, Etime, name)
FriendRequest(Member, Friend , Accepted, DateMet, FriendType, Experience, showFriends, NumRequests)
Host (HostID, **CouchID**)
Guest (GuestID)
StayReview (Testimony, TestimonyID, **MemID**, **HostID**, Guest Rating)
CouchRequest (RequestID, Arrival, Departure, Status, NSurfers, Personality, **GuestID**, **CouchID**, **HostID**)
Couch (CouchID, **HostID** , size, availability)
SearchHistory(HistoryID, **MemID** ,Availability, Gender)
Notification(NotificationID, **HistoryID**, **MemID**)
SearchReturn(ReturnId, **HistoryID**, **CouchID**)
Group (Gname, GroupID , Type)
GroupMembers(GroupID, **MemID**)
Post(PostID, content, **MemID**, **GroupID**)
Response (ResID, Content, **MemID**, **PostID**)
Administrator(AdminID)
Report(ReportID, Description, **GroupID**, **MemID**)

Part D PostgreSQL Definition

```
DROP SCHEMA IF EXISTS A3 CASCADE;
CREATE SCHEMA A3;
SET search_path TO A3;

DROP TABLE IF EXISTS Member CASCADE;
DROP TABLE IF EXISTS Invite CASCADE;
DROP TABLE IF EXISTS Event CASCADE;
DROP TABLE IF EXISTS FriendRequest CASCADE;
DROP TABLE IF EXISTS Host CASCADE;
DROP TABLE IF EXISTS Guest CASCADE;
DROP TABLE IF EXISTS StayReview CASCADE;
DROP TABLE IF EXISTS CouchRequest CASCADE;
DROP TABLE IF EXISTS Couch CASCADE;
DROP TABLE IF EXISTS SearchHistory CASCADE;
DROP TABLE IF EXISTS Notification CASCADE;
DROP TABLE IF EXISTS SearchReturn CASCADE;
DROP TABLE IF EXISTS Group CASCADE;
DROP TABLE IF EXISTS GroupMembers CASCADE;
DROP TABLE IF EXISTS Post CASCADE;
DROP TABLE IF EXISTS Response CASCADE;
DROP TABLE IF EXISTS Administrator CASCADE;
DROP TABLE IF EXISTS Report CASCADE;




CREATE TABLE Member(
MemId               INTEGER   PRIMARY KEY,
Email               VARCHAR(20)  NOT NULL,
Gender              VARCHAR(10)  NOT NULL,
BirthDate           VARCHAR(20)  NOT NULL,
RealName            VARCHAR(20)  NOT NULL,
Country                 VARCHAR(20) NOT NULL,
City                    VARCHAR(20) NOT NULL,
Interest                VARCHAR(500) NOT NULL,
CredibilityScore        VARCHAR(20) NOT NULL
);

CREATE TABLE Invite(
InviteId               INTEGER   PRIMARY KEY,
EMemId                 INTEGER   REFERENCES   Event(EMemId) ON
DELETE RESTRICT,
MemId                  INTEGER   REFERENCES   GroupMembers(MemId)
ON DELETE RESTRICT,
EventID                INTEGER    REFERENCES    Event(EventID) ON
DELETE RESTRICT
);

CREATE TABLE Event(
 EventID            INTEGER   NOT NULL,
 CreatorID          INTEGER   NOT NULL,
 EMemID             INTEGER   NOT NULL,
 description        VARCHAR(250)  NOT NULL,
 Edate              VARCHAR(20)   NOT NULL,
 location           VARCHAR(60)   NOT NULL,
 Etime              VARCHAR(20)   NOT NULL,
 location           VARCHAR(100)  NOT NULL,
 name               VARCHAR(20)   NOT NULL,
PRIMARY KEY(EventID,CreatorID,EMemID)
);



CREATE TABLE FriendRequest(

Member             INTEGER   NOT NULL,
Friend             INTEGER   NOT NULL,
Accepted          BOOLEAN   NOT NULL,
DateMet           VARCHAR(20)   NOT NULL,
FriendType        VARCHAR(20)   NOT NULL,
Experience        VARCHAR(100)  NOT NULL,
showFriends       BOOLEAN       NOT NULL,
NumRequests       INTEGER       NOT NULL
PRIMARY KEY(Member,Friend)
);

CREATE TABLE Host (
HostID            INTEGER   PRIMARY KEY,
CouchID           INTEGER   REFERENCES Couch(CouchID) ON
DELETE RESTRICT,
);

CREATE TABLE Guest(
GuestID          INTEGER   PRIMARY KEY
);

CREATE TABLE StayReview(
Testimony         VARCHAR(300)      NOT NULL,
TestimonyID       INTEGER           PRIMARY KEY,
MemID             INTEGER           REFERENCES Member(MemID)
ON DELETE RESTRICT,
HostID            INTEGER REFERENCES Host(HostID) ON DELETE
RESTRICT,
Guest Rating      INTEGER  NOT NULL
);

CREATE TABLE CouchRequest(
RequestID         INTEGER       PRIMARY KEY,
Personality       VARCHAR(300)  NOT NULL,
Arrival           VARCHAR(30)           NOT NULL,
Departure         VARCHAR(30)     NOT NULL,
Status            VARCHAR(20)     NOT NULL,
NSurfers            INTEGER        NOT NULL,
GuestID             INTEGER         REFERENCES Guest(GuestID) ON
DELETE RESTRICT,
CouchID                     INTEGER          REFERENCES
Couch(CouchID)  ON DELETE RESTRICT,
HostID                      INTEGER          REFERENCES  Host(HostID)
ON DELETE RESTRICT
);

CREATE TABLE Couch(
CouchID       INTEGER     PRIMARY KEY,
GuestID                     INTEGER    REFERENCES Guest(GuestID) ON
DELETE RESTRICT,
HostID                      INTEGER      REFERENCES  Host(HostID)  ON
DELETE RESTRICT,
Size                        INTEGER      NOT NULL,
Availability           VARCHAR(20)   NOT NULL
);

CREATE TABLE SearchHistory(
HistoryID    INTEGER    PRIMARY KEY,
MemID     INTEGER      REFERENCES Member(MemID) ON DELETE
RESTRICT,
Availability  VARCHAR(20) NOT NULL,
Gender     VARCHAR(15)   NOT NULL
);



CREATE TABLE Notification(
NotificationID    INTEGER   PRIMARY KEY,
HistoryID   INTEGER        REFERENCES SearchHistory(HistoryID) ON
DELETE RESTRICT,
MemID        INTEGER        REFERENCES Member(MemID) ON
DELETE RESTRICT
```

```
);

CREATE TABLE SearchReturn(
ReturnID    INTEGER     PRIMARY KEY,
HistoryID   INTEGER    REFERENCES SearchHistory(HistoryID) ON
DELETE RESTRICT,
MemID         INTEGER   REFERENCES  Member(MemID) ON DELETE
RESTRICT
);

CREATE TABLE Group (
GroupID INTEGER   PRIMARY KEY,
Gname      VARCHAR(20)   NOT NULL,
Type       VARCHAR(20)   NOT NULL
);

CREATE TABLE GroupMembers(
GroupID     INTEGER   PRIMARY KEY,
MemID      INTEGER   REFERENCES  Member(MemID) ON DELETE
RESTRICT
);

CREATE TABLE Post (
PostID  INTEGER    PRIMARY KEY,
Content VARCHAR(300)  NOT NULL,
MemID  INTEGER   INTEGER   REFERENCES  Member(MemID) ON
DELETE RESTRICT,
GroupID INTEGER    REFERENCES   GroupMembers(GroupID) ON
DELETE RESTRICT
);



CREATE TABLE Response (
ResID INTEGER     PRIMARY KEY,
Content          VARCHAR(300) NOT NULL,
MemID INTEGER     REFERENCES  Member(MemID) ON DELETE
RESTRICT,
PostID INTEGER    REFERENCES  Post(PostID)  ON DELETE
RESTRICT
);

CREATE TABLE Administrator(
AdminID     INTEGER          PRIMARY KEY
);

CREATE TABLE Report(
ReportID      INTEGER       PRIMARY KEY,
Description         VARCHAR(50),
GroupID      INTEGER       REFERENCES Group(GroupID) ON DELETE
RESTRICT,
MemID         INTEGER      REFERENCES Member(MemID) ON
DELETE RESTRICT
);
```