EECS 3421 Assignment 3

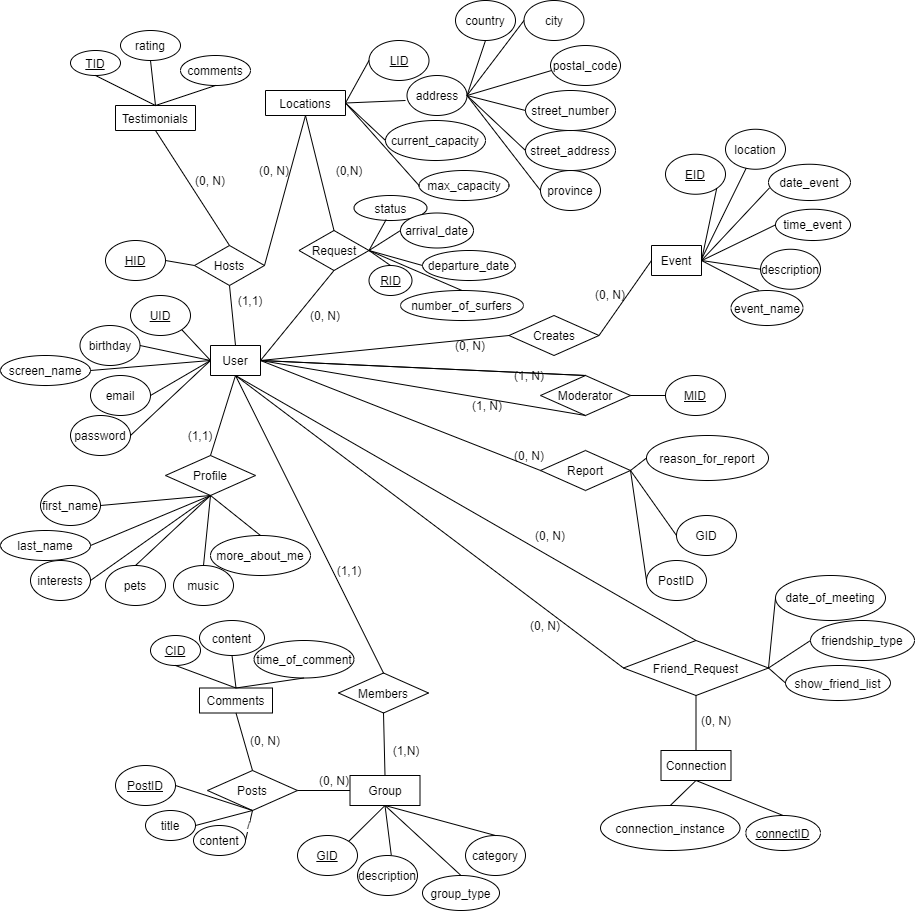
Name: Gavin Sit

Login: gavinsit

Student #: 215 043 870

**Part A**

* The user has only one email that they connect with the account when signing up which can change but system will allow only one email on file
* Guests can also be hosts and vice versa
* Users (both host and guest) will have a screen name to identify them since real name is optional information to include in profile
* All administrators receive the same reports which they can select which report to review and examine (when one admin reviews the report, it becomes unavailable to another admin to review)
* Hosts can have multiple locations where they are hosting surfers
* A user who is also a host has only one host profile (even if they have multiple locations)
* One location has at least 1 spot for surfers and up to n spots (depending on capacity of the location)
* User can only make one request for one location
* Surfers can only write testimonials of the host if their stay request has been accepted and the it is pass the arrival\_date (because they can still cancel before the arrival\_date and also haven’t started getting hosted so how would they know how the experience was)
* A group has at least one member (the creator of the group)
* Reports are made against either the group as a whole or an individual post in a group
* There is at least 1 moderator that moderates 0-N users (excluding themselves)
* Users can send and receive any number of friend requests
* When a friend request is sent, it is possible there are no connections
  + E.g. both users just join the app and add each other as friends
* A friend request can be made to one user once (but can cancel request and resend the request)
* A user can only be part of the same group once
* All booking requests made will be valid because the system will check before submitting the query
  + E.g. the number\_of\_surfers in request cannot exceed the maximum capacity when you add current\_capacity and number\_of\_surfers
* There can be more than one report made against a group/post (by different users)
* The search function is a query and therefore will not have a relation of its own
  + Filters for the search will be the WHERE condition
* There can only be one host in one location (cannot have multiple hosts with the exact same location/address)
* If the group creator/admin is deleted (deletes their account), the group isn’t deleted, ownership will be transferred in some way
* Reports are not deleted even if the group or post the report is against is deleted

**Part B**

**Part C**

**Related to User**

User (UID, birthday, screen\_name, email, password)

* Need a screen\_name as real name is optional

Profile (UID, first\_name, last\_name, interests, pets, music, more\_about\_me)

* Profile with foreign key UID from User

Friend\_Request (UID, UID2, date\_of\_meeting, friendship\_type, show\_friend\_list)

* UID is requesting UID2 to be a friend

Connection (connectID, UID, UID2, connection\_instance)

* UID, and UID2 could have met on more than one instance

**Related to Hosting and Surfing**

Host (HID, UID)

* Each user who chooses to host is given a unique HID to identify the host related relations

Location (LID, HID, current\_capacity, max\_capacity city, postal\_code, street\_number, street\_address, province, country)

Request (RID, UID, LID, arrival\_date, departure\_date, number\_of\_surfers, status)

* UID is the user requesting to stay at location LID

Testimonials (TID, UID, tohost, rating, comments)

* User UID is giving tohost (HID) a rating

**Related to Groups**

Group (GID, UID, description, group\_type, category)

* UID created group GID

Members (UID, GID)

* User with UID is a part of group with GID

Posts (PostID, GID, title, content)

* Unique key assigned to each group as well as which group it belongs to

Comments (CID, PostID, content, time\_of\_comment)

* Comments have unique id as well as which post it is from

**Related to Events**

Event (EID, UID, event\_name, location, date\_event, time\_event, description)

* User UID creates event EID

**Related to Administrators**

Moderator (MID, UID)

* Moderators are assigned MID

Report (RID**,** GID, PostID, reason\_for\_report)

**Part D**

create table User (

UID int primary key,

birthday date not null,

screen\_name varchar (50) not null,

email varchar (100) not null,

password varchar(20) not null

);

create table Profile (

UID int,

first\_name varchar(50),

last\_name varchar(50),

interests varchar(500),

pets varchar (500),

music varchar (500),

more\_about\_me varchar(500),

foreign key (UID) references User(UID)

on delete cascade

);

create table Friend\_Request(

UID int,

UID2 int,

date\_of\_meeting date,

friendship\_type char(20) not null,

show\_friend\_list boolean,

foreign key (UID) references User(UID)

on delete cascade,

foreign key (UID2) references User(UID)

on delete cascade

);

create table Connection (

connectID int primary key,

UID int,

UID2 int,

connection\_instance varchar(100),

foreign key (UID) references User(UID)

on delete cascade,

foreign key (UID2) references User(UID)

on delete cascade,

);

create table Host (

HID int primary key,

UID int,

foreign key (UID) references User(UID)

on delete cascade,

);

create table Location(

LID int primary key,

HID,

current\_capacity int not null,

max\_capacity int not null,

city varchar(30) not null,

postal\_code char(6),

street\_number int not null,

street\_address varchar(30) not null,

province varchar (30) not null,

country varchar(30) not null,

foreign key (HID) references Host(HID)

on delete cascade

);

create table Request (

RID int primary key,

UID int,

LID int,

arrival\_date date not null,

departure\_date date not null,

number\_of\_surfers int not null,

status char (8) not null,

foreign key (UID) references User(UID)

on delete cascade,

foreign key (LID) references Location(LID)

on delete cascade

);

create table Testimonials (

TID int primary key,

UID int,

tohost int,

rating int,

comments varchar (500),

foreign key (UID) references User(UID)

on delete cascade,

foreign key (tohost) references Host(HID)

);

create table Group (

GID int primary key,

UID int,

description varchar(200),

group\_type varchar(20) not null,

category varchar (20) not null,

foreign key (UID) references User(UID) --dont delete group if UID is deleted

);

create table Members (

UID int,

GID int,

foreign key (UID) references User(UID)

on delete cascade,

foreign key (GID) references Group(GID)

on delete cascade,

primary key (UID, GID)

);

create table Posts(

PostID int primary key,

GID int,

title varchar(30) not null,

content varchar (500) not null,

foreign key (GID) references Group(GID)

on delete cascade,

);

create table Comments (

CID int primary key,

PostID int,

content not null,

time\_of\_comment timestamp not null,

foreign key (PostID) references Posts(PostID)

on delete cascade,

);

create table Event (

EID int primary key,

UID int,

event\_name varchar(50) not null,

location varchar(200) not null,

date\_event date not null,

time\_event time not null,

description varchar (500) not null,

foreign key (UID) references User(UID)

on delete cascade,

);

create table Moderator (

MID int primary key,

UID int,

foreign key (UID) references User(UID)

on delete cascade,

);

create table Report (RID

RID int primary key,

GID int,

PostID int,

reason\_for\_report varchar(200),

foreign key (GID) references Group(GID),

foreign key (PostID) references Posts(PostID)

);