Department of Electrical and Computer Engineering
University of Puerto Rico Mayagüez Campus
CIIC 4060/ICOM 5016 – Introduction to Database Systems

Spring 2018 Term Project
Social Messaging Application Phase I:
Conceptual Design

Gladymar O'Neill
Kristalys Ruiz
Cristian Duque
March 28, 2018

User: Refers to every user in the UI.

Attributes:	uid: user id, primary key.	[serial]
	name: first_name and last_name composite attribute.	[varchar(30)]
	phone: phone number a user uses to register in the system.	[varchar(11)]
	email: email used to register to the system.	[varchar(40)]
	password: password for the user login	[varchar(10)]
	is_active: will determine if the user is currently active.	[boolean]

Relationships of **User:**

Owns: A user is mapped to **Chats**. A user can be the administrator of a chat

Participates: A user is mapped to **Chats.** A user participates in a particular chat.

Owns: A user is mapped to **Contact_list**. A user owns a contact list of users of the application.

Sends: A user is mapped to **Messages**. A user sends a message to a chat.

Liked by: A user is mapped to **Messages**. A user can like a message.

Disliked by: A user is mapped to **Messages**. A user can dislike a message.

Contact_list: Refers to all the contacts (users) a particular user has.

Attributes:	lid: refers to the id of the contact list.	[integer]
	owner: refers to the owner of the contact list.	[integer]

Relationships of **Contact_list**:

***Has*:** Relationship between contact list and user that establish that a contact list has one or more users.

Chats: Relation to maintain a record of all of the chats that have been created in the system

Attributes:	cid: chat id.	[serial]
	chat_name: name of the chat.	[varchar(30)]
	owner: refers to the user id in the User table of the person that created the chat	[integer]

Relationships of **Chats**

Contains: A chat list is mapped with **Messages** to refer to the number of messages the chat has.

Message: refers to every message send in any chat

Attributes:	mid: message id.	[serial]
	text: the body of the message.	[varchar(200)]
	sender: refers to the user id who sent the message.	[integer]
	date: the date the messages was sent.	[date]
	time: the time the message was sent.	[time]
	chat_sent_to_id: refers to the chat the message was sent to.	[integer]

Relationships of **Message**:

Reply: Relationship between two messages, the original message and its reply.

Attributes:	reply_mid: id of the reply message.	[serial]
	replying_to_mid: id of the message that is being replied.	[integer]

Contains: A message is mapped to **Hashtags.** A message can contain a particular hashtag.

Hashtags: Relation that states all the hashtags used in every message on the system in any chat. This will be used to manage trending topics in the web application

Attributes:	hashid: hashtag id	[serial]
	hashname: the text after the # in a hashtag	[varchar(200)]

Tables Mapping

Entities:

create table User(uid serial primary key, firstname varchar(30), lastname varchar(30), phone long, email varchar(30), password varchar(10))

create table Chat(cid serial primary key, chatname varchar(30), owner long foreign key references user(uid))

create table Message(mid serial primary key, text varchar(200), sender long foreign key references user(uid), dateofmessage date, timeofmaessage time, chatsendto long foreign key references chat(cid))

create table ContactList(clid serial primary key, owner long foreign key references Use(uid))

create table Hashtag(hid serial primary key, hashname varchar(30))

Relations:

create table ReplyRelation(rrid primary key(sendedmessage,replytomessage), sendedmessage long foreign key references Message(mid), replytomessage long foreign key references Message(mid))

create table ParticipatesRelation(prid primary key(useronchat, chatuserbelong), useronchat long foreign key references user(uid), chatuserbelong long foreign key references chat(cid))

create table DislikeRelation(drid primary key(messagedisliked, userdislikes), messagedisliked long foreign key references Message(mid), userdislikes long foreign key references user(uid))

create table LikedRelation(lrid primary key(messageliked, userliskes),messageliked long foreign key references Message(mid), userlikes long foreign key references user(uid))

create table ContainsHashRelation(chrid primary key(messagewithhash, hashincluded), messagewithhash long foreign key references Message(mid), hashincluded long key references Hashtag(hid))