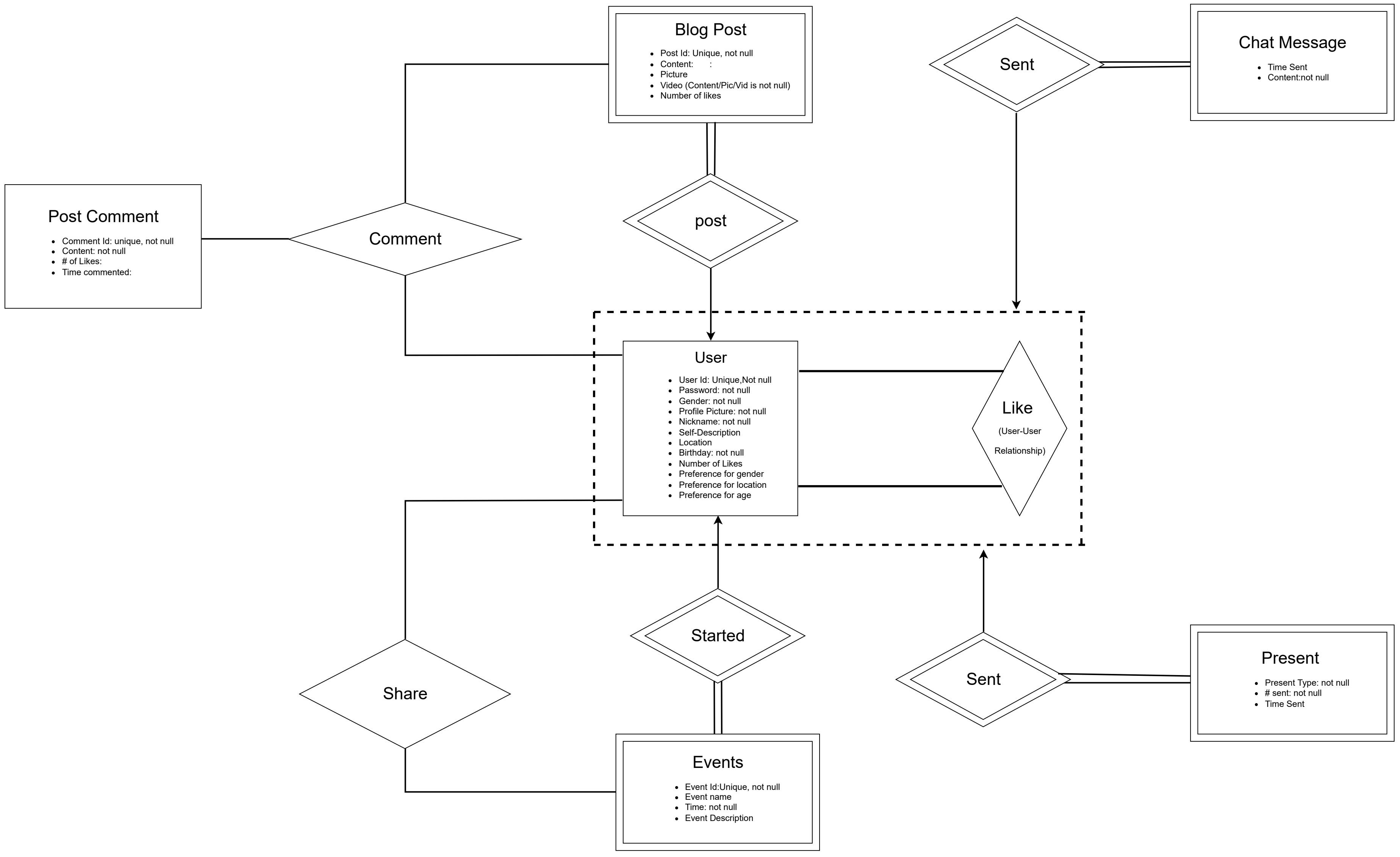
Description：

- This database design is for a dating app which allows users to like/dislike, chat, post, setup events and send event invite and interact with other users.
- For example, when a user registered for an account, he/she first need to enter several required attributes such as gender, preference, birthdate, what kind of relationship they are looking for and other optional attributes such as location, education, interests and etc. He/she is also required to upload a profile picture. For example, a 23 years old female user's profile can have nickname as Meow who lives Long Island City and prefers to date men who live in Manhattan (i.e. User (121, 1Aa123456, 0, Meow, null, Long Island City, 04-26-1999, 13, 1, Manhattan, null)). After signing up, the he/she can choose to like/dislike other users. He/she can also send presents (such as roses & hearts) to the other person which would automatically labelled as liked. Once a like is send, the liked person can start the conversation with he/she. If they are having a good time talking with each other and decide to go out for a date. He/she can setup an event (having lunch, watching movie, go to museum…) and send an invite to the other person. He/she can also post pictures and videos for all other users to see and other users can also make comments. Our database entities include users, chats, blog posts, presents, comments and events. Attributes for users, for example, include birthday, gender, nickname, password and etc. One of the constraints is that there should be only one chat between two users. Also, blog posts and events are weak entities of users. We have also created an aggregation called Like which represents pairs of users A and B (i.e. 121 & 131) where A have liked/sent present to B. Once such pair is created, B can start a chat with A.
- We will create our own data.
- Contingency plan: we can delete some social functionality of the app by deleting post, comment and present entities, which leaves only 4-5 entities.


Blog Post
Post Id: Unique, not null
Content: :
Picture
Video (Content/Pic/Vid is not null)
Number of likes
Sent
Chat Message
Time Sent
Content:not null
Post Comment
Comment Id: unique, not null
Content: not null
# of Likes:
Time commented:
Comment
post
User
User Id: Unique,Not null
Password: not null
Gender: not null
Profile Picture: not null
Nickname: not null
Self-Description
Location
Birthday: not null
Number of Likes
Preference for gender
Preference for location
Preference for age
Like
(User-User
Relationship)
Started
Share
Sent
Present
Present Type: not null
# sent: not null
Time Sent
Events
Event Id:Unique, not null
Event name
Time: not null
Event Description

SQL Schema:

```
CREATE TABLE Users(
	uid int,
	password text NOT NULL,
	gender int NOT NULL,
	pic: BLOB NOT NULL,
	nickname text NOT NULL,
	self_description text,
	location text,
	birthday date NOT NULL,
	nlikes int,

//p_   stands for preference
	p_ gender int,
	p_location text,
	p_age int,

	PRIMARY KEY (uid)
)

CREATE TABLE Likes(
// a tuple (A,B) represents A likes B and B can send message to A (one need to be liked before
//sending message to the other)
	A_uid int NOT NULL REFERENCES Users,
	B_uid int NOT NULL REFERENCES Users,

	PRIMARY KEY (A_uid, B_uid)
)

CREATE TABLE Chat_Messages(
	sender int NOT NULL REFERENCES Users,
	receiver int NOT NULL REFERENCES Users,
	time_sent datetime NOT NULL,
	content text NOT NULL,

	PRIMARY KEY (sender, receiver, time_sent),
	FOREIGN KEY (receiver, sender) REFERENCE Like_Chat
		ON DELETE CASCADE
)
```

```
CREATE TABLE Posts(
        pid int,
        uid int,
        content text,
        pic BLOB,
        vid BLOB,
        nlikes int,

        PRIMARY KEY (pid),
        CHECK (content is not null and content is not ‘’) or pic is not null or vid is not null,
        FOREIGN KEY (uid) REFERENCE User
                ON DELETE CASCADE

)

CREATE TABLE Comments(
        cid int,
        content text NOT NULL,
        nlikes int,
        time_comment datetime,

        PRIMARY KEY (cid)
)

CREATE TABLE Comment_Relations(
        cid int NOT NULL REFERENCE Comment,
        uid int NOT NULL REFERENCE User,
        pid int NOT NULL REFERENCE Post,

        PRIMARY KEY (cid, uid, pid)
)


CREATE TABLE Presents_Sent(
        P_type int,
        time_sent datetime NOT NULL,
        num_sent int NOT NULL,
        sender int NOT NULL REFERENCES Users,
        receiver int NOT NULL REFERENCES Users,

        PRIMARY KEY(sender, receiver, P_type, time_sent),
        FOREIGN KEY(sender, receiver) REFERENCE Like_Chat
                ON DELETE CASCADE
)



CREATE TABLE Started_Events (
	eid int serial,
	uid int,
	name text NOT NULL,
	description text,
	date datetime NOT NULL,

	PRIMARY KEY(eid),
	FOREIGN KEY(uid) REFERENCE User
		ON DELETE CASCADE
)

CREATE TABLE Shared_Events(
	eid int NOT NULL REFERENCE Started_Events,
	invitee int NOT NULL REFERENCE User,

	PRIMARY KEY(eid, invitee)
)
```