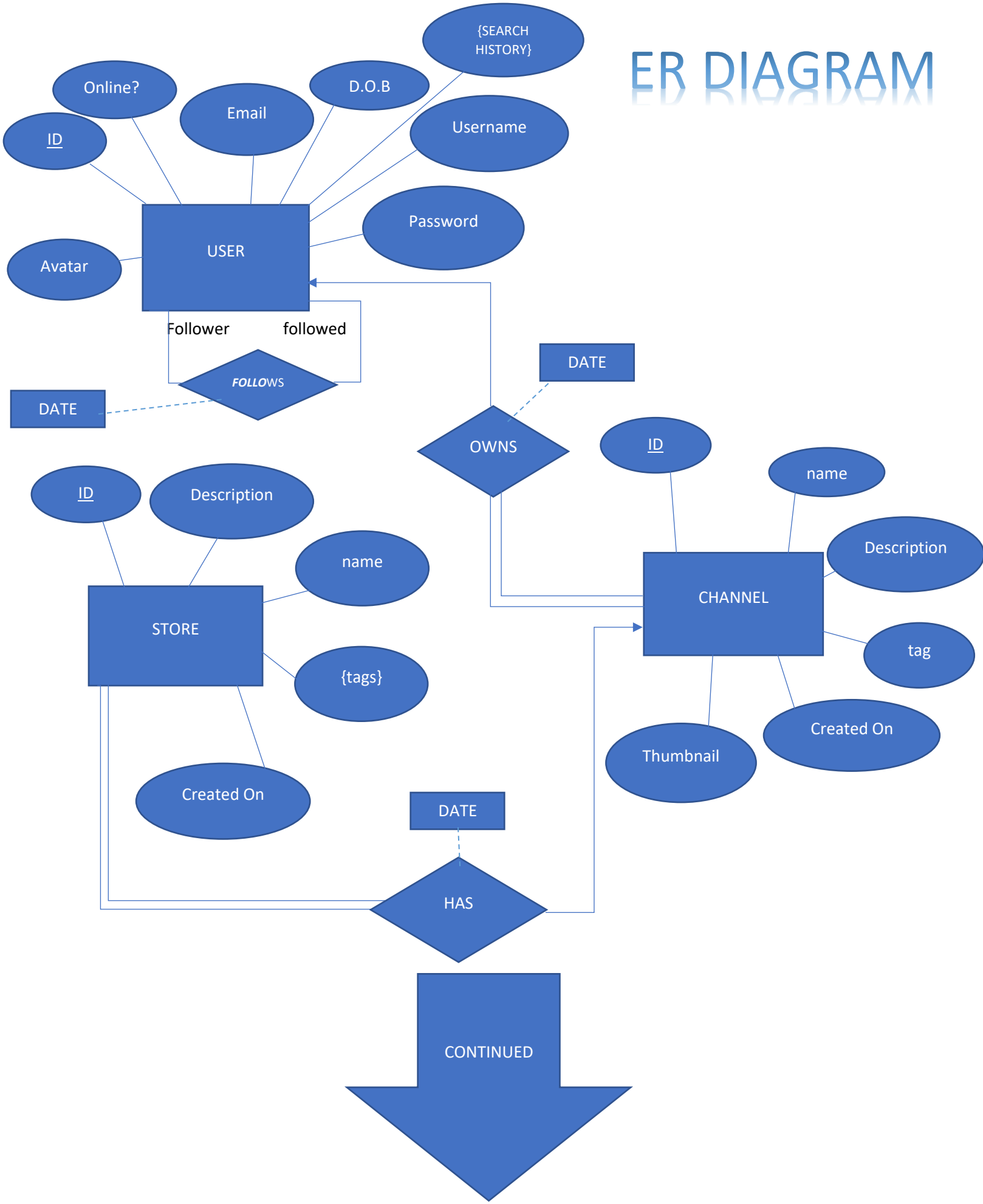
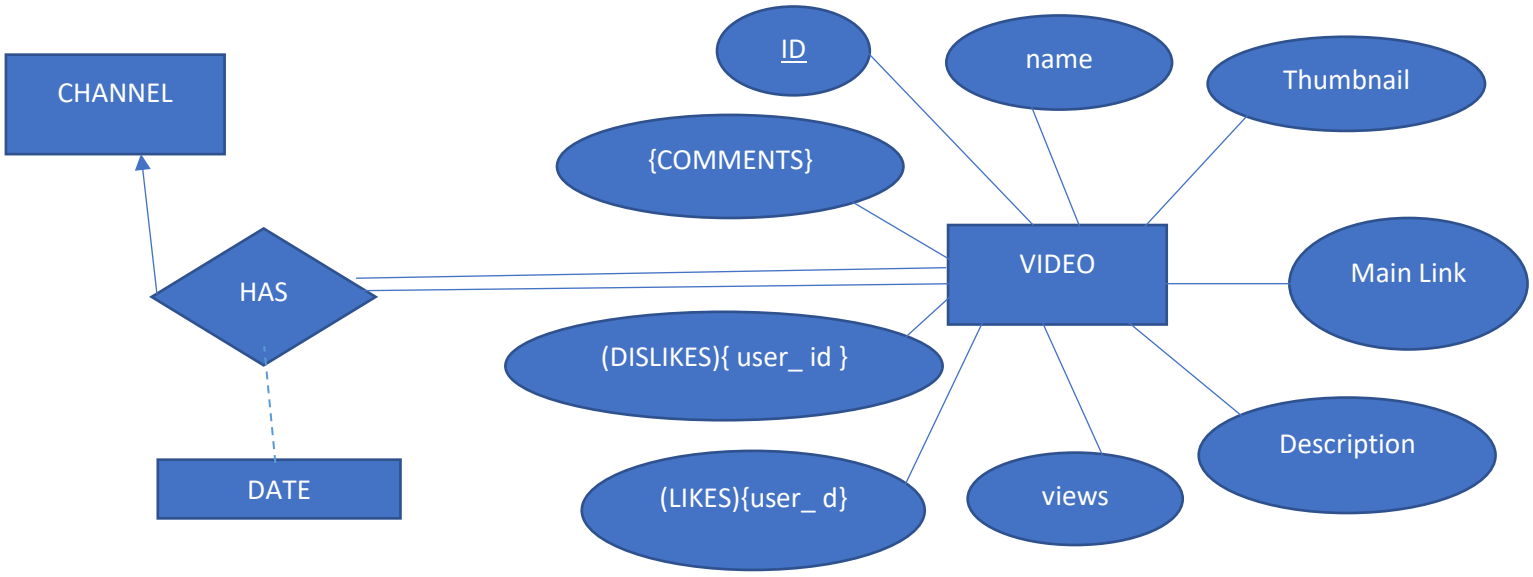
TWICH RETRO

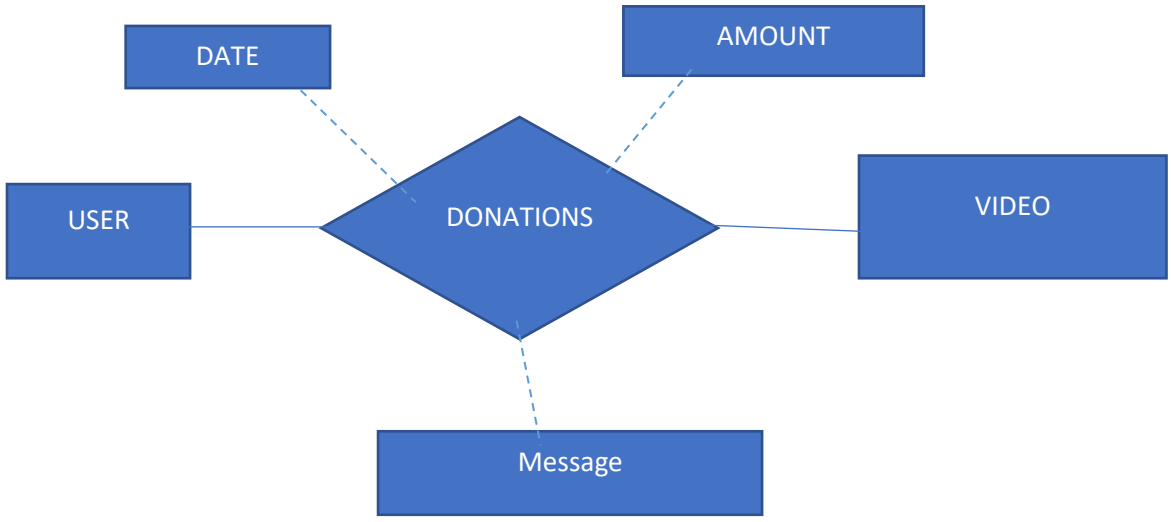
TEAM MEMBERS:

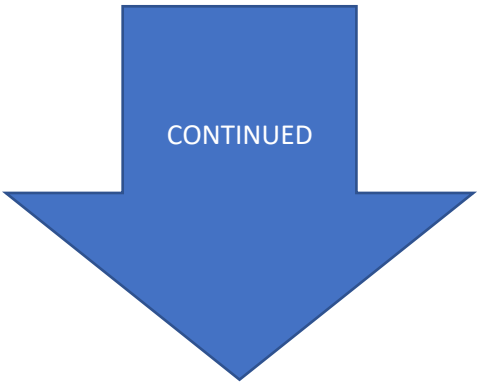
NAME	REG.NO	ROLL.NO	SECTION	CONRIBUTION
ROLYN JASE MACHADO	180953014	08	CCE-A	ER, SCHEMA
ABHIK RAY	180953026	14	CCE-A	Normalization
ADITI PALA	180953035	12	CCE-B	UI
AKSHAY MHATRE	180953021	13	CCE-A	Stored Procedures

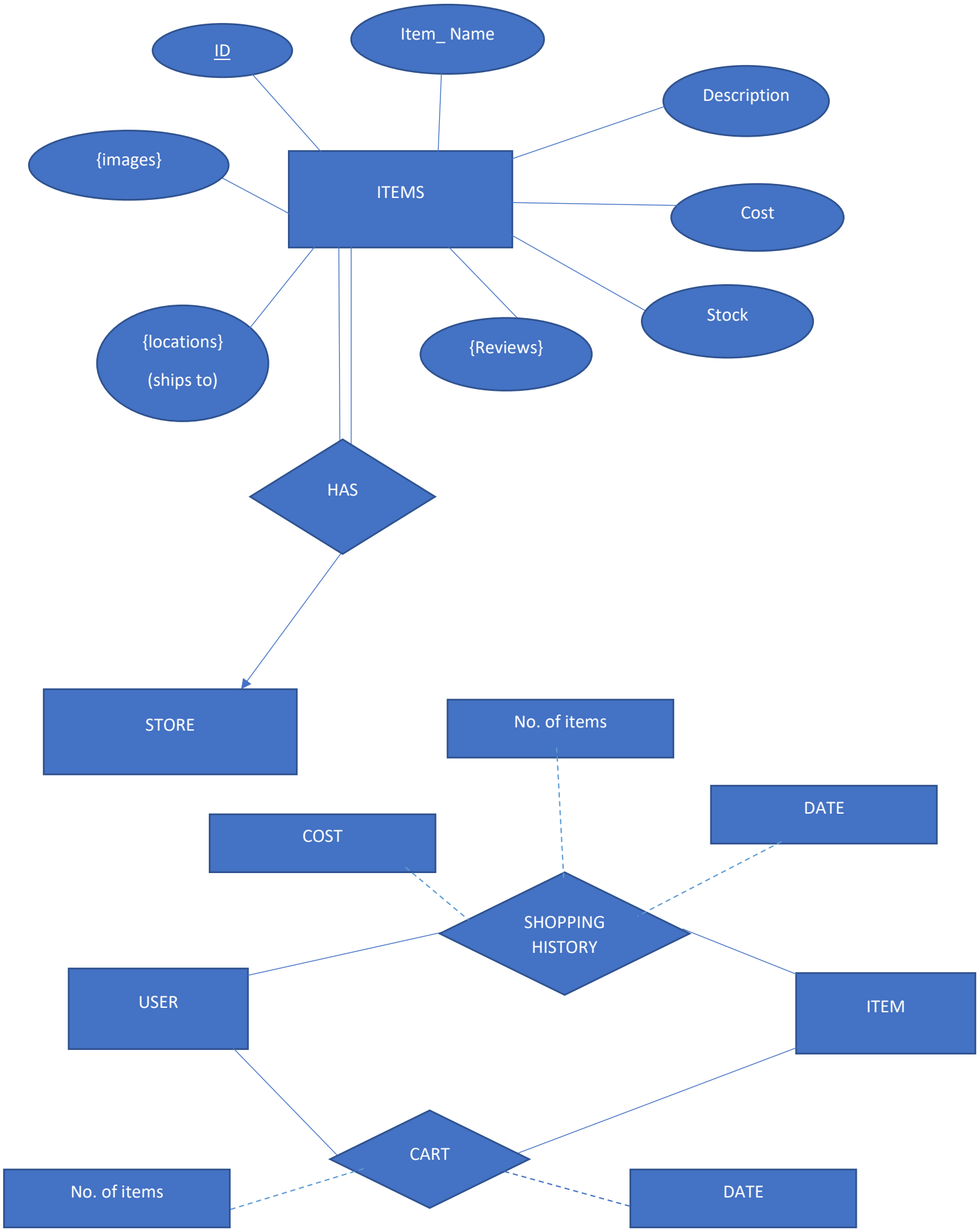
ER DIAGRAM

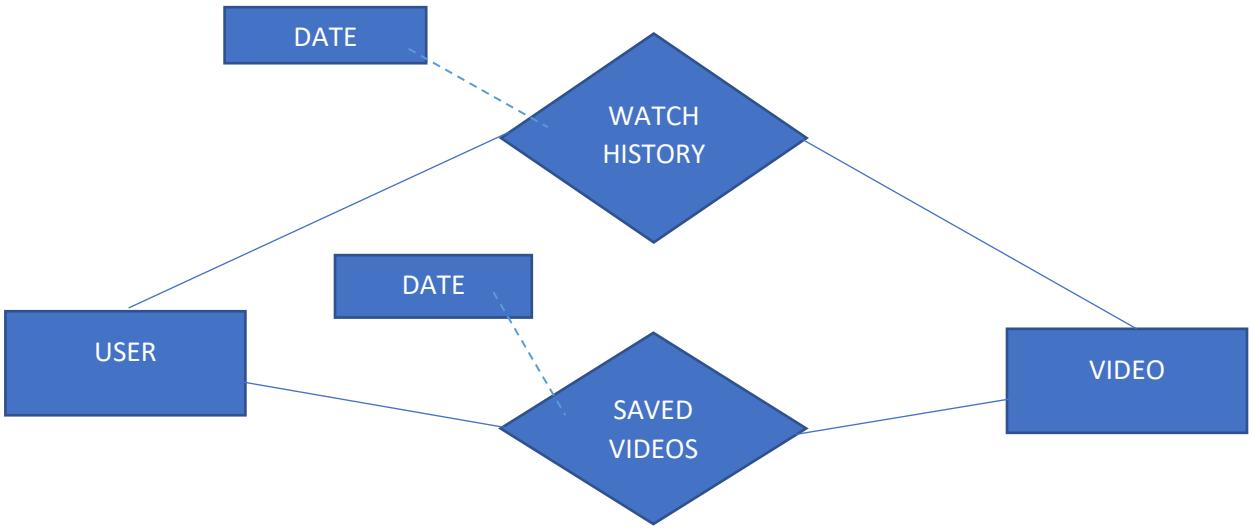

{SEARCH HISTORY}
Online?
D.O.B
Email
Username
ID
USER
Password
Avatar
Follower
followed
FOLLOWS
DATE
DATE
OWNS
ID
name
ID
Description
Description
name
CHANNEL
STORE
tag
{tags}
Thumbnail
Created On
Created On
DATE
HAS
CONTINUED


CHANNEL
ID
name
Thumbnail
{COMMENTS}
HAS
VIDEO
Main Link
(DISLIKES){ user_ id }
DATE
Description
(LIKES){user_ d}
views

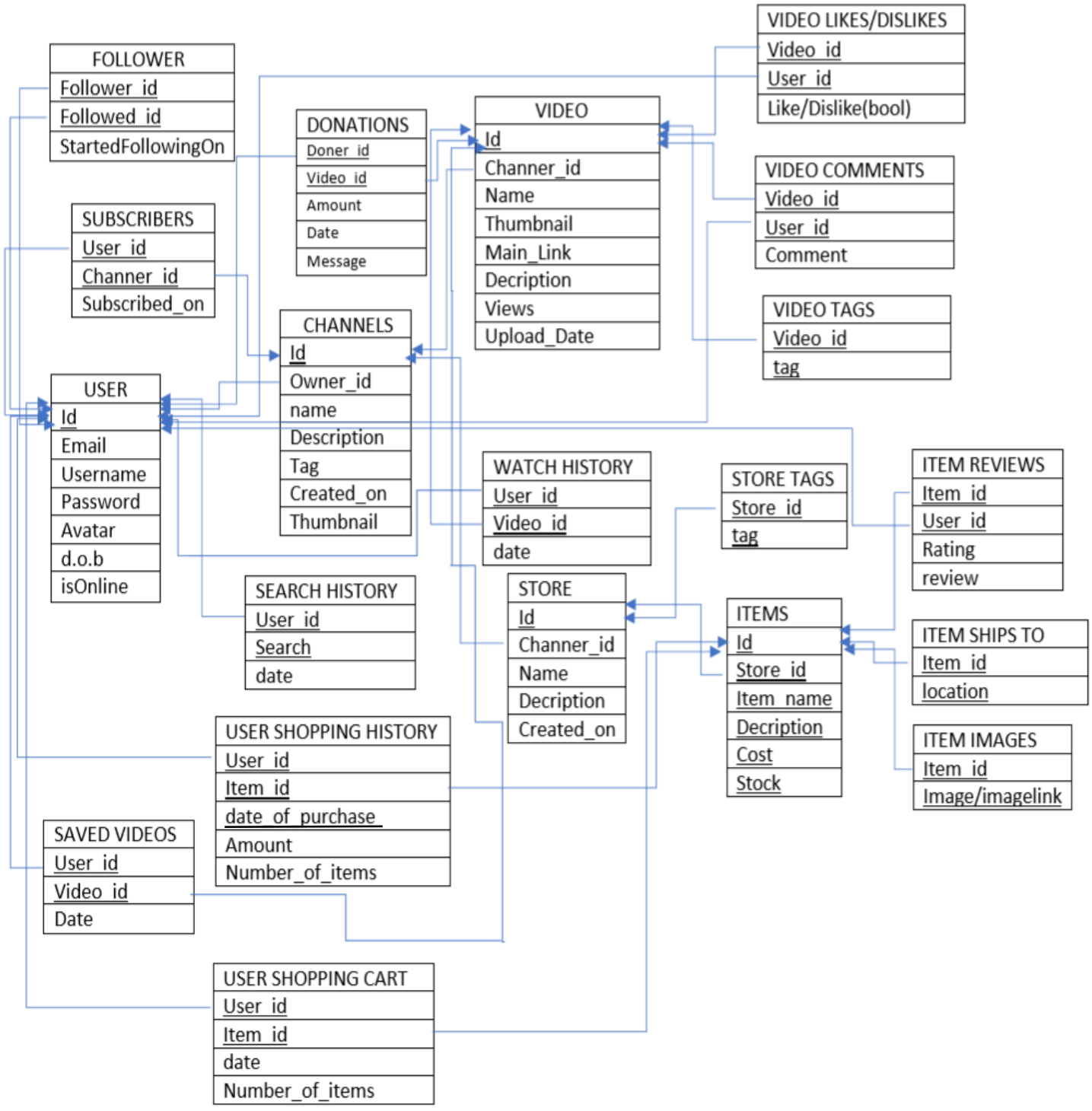

DATE
AMOUNT
USER
DONATIONS
VIDEO
Message


CONTINUED


ID
Item_ Name
Description
{images}
ITEMS
Cost
Stock
{locations}
(ships to)
{Reviews}
HAS
STORE
No. of items
DATE
COST
SHOPPING
HISTORY
USER
ITEM
CART
No. of items
DATE


DATE
WATCH HISTORY
DATE
USER
SAVED VIDEOS
VIDEO

SCHEMA


FOLLOWER
Follower_id
Followed_id
StartedFollowingOn
SUBSCRIBERS
User_id
Channer_id
Subscribed_on
USER
Id
Email
Username
Password
Avatar
d.o.b
isOnline
DONATIONS
Doner_id
Video_id
Amount
Date
Message
CHANNELS
Id
Owner_id
name
Description
Tag
Created_on
Thumbnail
VIDEO
Id
Channer_id
Name
Thumbnail
Main_Link
Decription
Views
Upload_Date
VIDEO LIKES/DISLIKES
Video_id
User_id
Like/Dislike(bool)
VIDEO COMMENTS
Video_id
User_id
Comment
VIDEO TAGS
Video_id
tag
WATCH HISTORY
User_id
Video_id
date
SEARCH HISTORY
User_id
Search
date
STORE
Id
Channer_id
Name
Decription
Created_on
STORE TAGS
Store_id
tag
ITEMS
Id
Store_id
Item_name
Decription
Cost
Stock
ITEM REVIEWS
Item_id
User_id
Rating
review
ITEM SHIPS TO
Item_id
location
ITEM IMAGES
Item_id
Image/imagelink
USER SHOPPING HISTORY
User_id
Item_id
date_of_purchase
Amount
Number_of_items
SAVED VIDEOS
User_id
Video_id
Date
USER SHOPPING CART
User_id
Item_id
date
Number_of_items

FUNCTIONAL DEPENDENCIES

USER: ID->Email Username Password Avatar d.o.b isOnline. (BCNF)

CHANNEL: ID->Owner_id name description Tag created_on Thumbnail (BCNF)

STORE: ID->Channel_id Name Description Created_on (BCNF)

VIDEO: ID->Channel_id Name Thumbnail Main_link Description Views Upload_Date (BCNF)

ITEMS: ID->Store_id Item_name Description Cost Stock. (BCNF)

SUBSCRIBERS: User_id Channel_id ->Subscribed_on (BCNF)

FOLLOWER: Follower_id Followed_id->StartedFollowingOn (BCNF)

DONATIONS: Doner_id Video_id->Amount Date Message (BCNF)

VIDEOLD: Video_id User_id->Like/Dislike (BCNF)

VIDEO_COMMENTS: Video_id User_id->Comment (BCNF)

VIDEO_TAGS: Video_id tag->Video_id tag (BCNF)

STORE_TAGS: Store_id tag->Store_id tag (BCNF)

ITEM_REVIEW: Item_id User_id->rating review (BCNF)

Item_SHIPS_TO: Item_id location->Item_id location (BCNF)

SEARCH_HISTORY: User_id Searh->date (BCNF)

WATCH HISTORY: User_id Video_id -> date (BCNF)

USER_SHOPPING HISTORY: User_id Item_id date->Amount Number_of_items (BCNF)

USER_SHOPPING CART: User_id Item_id ->date Number_of_items (BCNF)

ALL THE TABLES ARE IN BCNF; THEREFORE, THE DATABASE IS IN BCNF

STORED PROCEDUERES

1: Login:

```
CREATE OR REPLACE FUNCTION Login ( Username IN Text, Password IN Text )
RETURN NUMBER
AS
FOUND INTEGER;
BEGIN
SELECT count(*) INTO FOUND
FROM User
Where User.username = Username && User.password == password;
IF (FOUND > 0)
        THEN
        UPDATE TABLE User SET IsOnline = 1 WHERE
        User.username=Username&&User.password=password;
        RETURN 1;
ELSE
        SELECT count(*) INTO FOUND
        FROM User
        Where User.username = Username;
        IF (FOUND > 0)
        THEN RETURN 2;
        ELSE RETURN 0;
        END IF;
END IF;
END;
```

2: Log Out:

```
CREATE OR REPLACE FUNCTION Logout ( UserId IN INTEGER)
RETURN NUMBER
AS
BEGIN
UPDATE TABLE User SET IsOnline = 0 WHERE User.Id = UserId;
RETURN 1;
END;
```

3: GetId:

```
CREATE OR REPLACE FUNCTION getId (Username IN text, Password IN text)
RETURN INTEGER
AS
Id INTEGER;
BEGIN
SELECT User.Id INTO Id FROM User
WHERE User.username=Username&&User.password=Password;
RETURN Id;
END;
```

4: GetChannels:

```
CREATE OR REPLACE FUNCTION getChannels (UserId IN INTEGER)
RETURN VARCHAR2
AS
ChannelList VARCHAR2(20);
BEGIN
SELECT Channels.Id, Channels.Owner_id INTO ChannelList FROM Channels
WHERE Channel.Owner_id=UserId;
RETURN ChannelList;
END;
```

5: GetChannelId:

```
CREATE OR REPLACE FUNCTION getChannelId (UserId IN INTEGER, ChannelName IN VARCHAR)
RETURN INTEGER
AS
ChannelID INTEGER;
BEGIN
SELECT Channels.Id  INTO ChannelID FROM Channels
WHERE Channel.Owner_id=UserId && Channel.name=ChannelName;
RETURN ChannelID;
END;
```

6: GetVideos:

```
CREATE OR REPLACE FUNCTION getVideos (ChannelId IN INTEGER)
RETURN VARCHAR2
AS
VideoList VARCHAR2(20);
BEGIN
SELECT Video.Id, Video.name INTO VideoList FROM Video
WHERE Vider.Channel_id=ChannelId;
RETURN VideoList;
END;
```

7:GetLikes:

```
CREATE OR REPLACE FUNCTION getLikes (VideoId IN INTEGER)
RETURN NUMBER
AS
LIKES NUMBER;
BEGIN
SELECT COUNT(*) INTO LIKES FROM VideoLD WHERE VideoLD.Video_id=VideoId && LD = 1;
RETURN LIKES;
END;
```

7:GetDisLikes:

```
CREATE OR REPLACE FUNCTION getDisLikes (VideoId IN INTEGER)
RETURN NUMBER
AS
DISLIKES NUMBER;
BEGIN
SELECT COUNT(*) INTO DISLIKES FROM VideoLD WHERE VideoLD.Video_id=VideoId && LD = 0;
RETURN DISLIKES;
END;
```

8:Search:

```
//using cursor
CREATE OR REPLACE FUNCTION Search (SearchText IN VARCHAR2)
RETURN SYS_REFCURSOR
AS
SearchResult SYS_REFCURSOR;
BEGIN
OPEN SearchResult FOR SELECT * FROM Video,VideoTags Where Video.id=VideoTags.Video_id
&& VideoTags.tag like *SearchText*;
RETURN SearchResult;
END;
```

9: GetVideoComments:

```
CREATE OR REPLACE FUNCTION getVideoCommnets (VideoId IN INTEGER)
RETURN VARCHAR2;
AS
Comments VARCHAR2;
BEGIN
SELECT User_id,Comment INTO Comments from VideoComments WHERE
VideoComments.Video_id=VideoId;
RETURN Commnets;
```

//NESTED QUERIES

getUserIDFromItemId:

```
CREATE OR REPLACE FUNCTION getUserIDFromitemid (itemID IN INTEGER)
RETURN INTEGER;
AS
USERID INTEGER;
BEGIN
SELECT unique User_id INTO USERID FROM STORE
WHERE STORE.id=(SELECT Store_id FROM ITEMS WHERE id =itemID);
RETURN USERID;
```

getUserIDFromVideoID:

```
CREATE OR REPLACE FUNCTION getUserIDFromVideoid (VideoID IN INTEGER)
RETURN INTEGER;
AS
USERID INTEGER;
BEGIN
SELECT unique User_id INTO USERID FROM CHANNELS
WHERE CHANNELS.id=(SELECT Channel_id FROM VIDEO WHERE id =VideoID);
RETURN USERID;
```

```
//TRIGGERS
CREATE OR REPLACE TRIGGER password
BEFORE INSERT OR UPDATE OF password
ON Users
BEGIN
IF (:new.password not like *[A-Z]* OR :new.password not like *[0-9]* OR LENGTH(:new.password < 8 ) )
THEN
RAISE_APPLICATION_ERROR(-20000,'Invalid Password');
END IF;
END;
```

```
CREATE OR REPLACE TRIGGER DeleteUser
BEFORE DELETE OF Id
ON Users
BEGIN
DELETE FROM Channels Where Channels.owner_id = :old. id;
DELETE FROM Subscribers Where Subcribers.User_id = :old. id;
DELETE FROM Follower Where Follower.Follower_id = :old. id;
```

```
DELETE FROM Donations Where Donations.doner_id = :old. id;
DELETE FROM SearchHistory Where User_id = :old. id;
DELETE FROM User_Cart Where User_id = :old. id;
DELETE FROM SavedVideos Where User_id = :old.id;
END;


CREATE OR REPLACE TRIGGER DeleteVideo
BEFORE DELETE OF Id
ON Video
BEGIN
DELETE FROM VIDEOLK Where Video_id=:old. id;
DELETE FROM Comments Where Video_id=:old. id;
DELETE FROM VideoTags Where Video_id=:old. id;

END;


CREATE OR REPLACE TRIGGER DeleteChannels
BEFORE DELETE OF Id
ON Channels
BEGIN
DELETE FROM Video where Channel_id=:old.id;
```

```
END;
```

UI IMPLEMENTATION

CONNECTING FRONT END (UNITY) WITH SQLITE DATABASE

```
using UnityEngine;
using Mono.Data.Sqlite; //Unity library for handling database functions.
using System.Data;
using UnityEngine.UI;

public class testdb : MonoBehaviour
{
    /*
        ->Given below is an EventListener;
        ->An EventListener can register itself to any number of events (of the same type);
        ->//in this implementation, it is directly called using the built in Unity UI;( dont need
to know this);
        ->When an event(like login button pressed,etc) is triggered,
          all the Listeners registered to that event will be invoked;
        ->And all the required operations(like retrieving data from the
database,Inseting,Deleting,etc) can be done
          the event listener;
        ->Below is am implementation which shows exactly how this done.
    */
    void AnyEventListner()
    {
        /*In our implementation the database is stored locally inside the application. But this
also works
         if the database is stored on a remote server . In the later case a TCP connection will be
established
        with the server and the database can be accessed as if it was stored locally*/



        string conn = "URI=file:" + Application.dataPath + "databasename.db";

        /*Path to database.
        (Application.dataPath gives the path of the resource folder within windows where the
database is stored)*/

        IDbConnection dbconn;
        dbconn = (IDbConnection)new SqliteConnection(conn);
        /*Establish  a connection with the database*/

        dbconn.Open();
        /*Open connection to the database.*/


        IDbCommand dbcmd = dbconn.CreateCommand();
        /*creates a new databaseCommand object.See documentation for more details.(link provided
below)*/
```

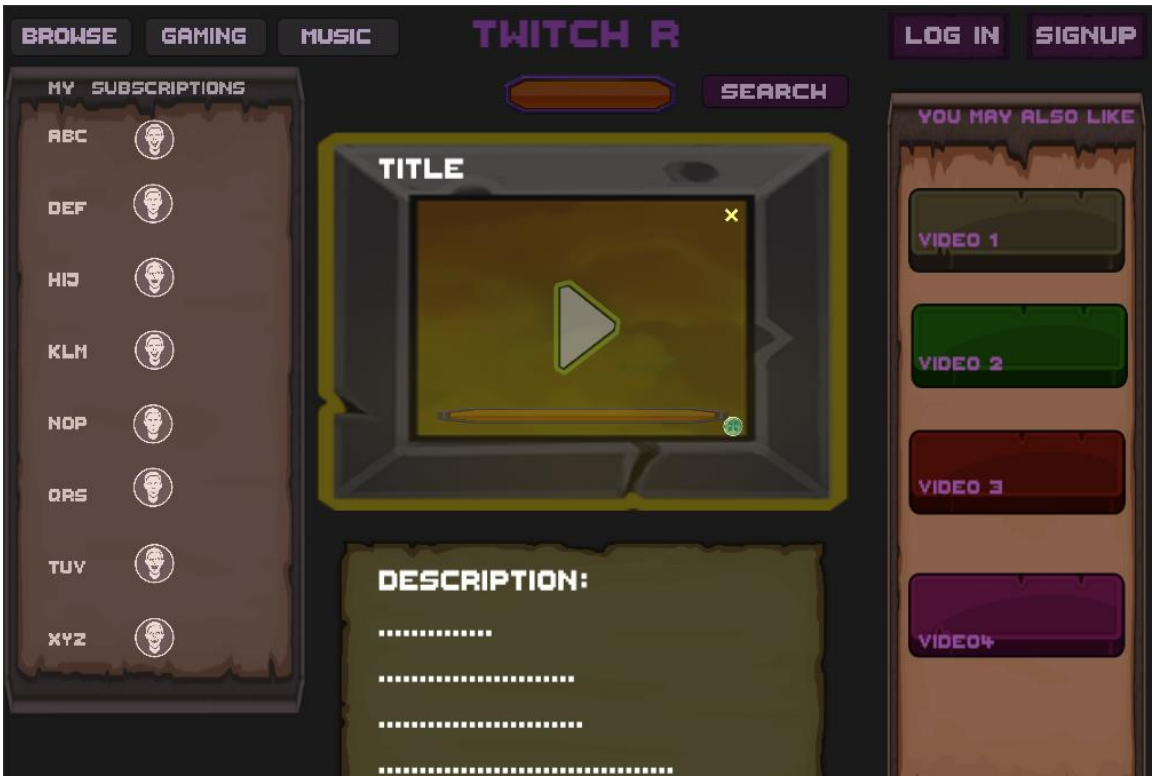
```
        string sqlQuery = "SELECT _id,_userName,_password " + "FROM User";
        /*This Query can be any valid sql statement( Select,insert,delete,update.SQLITE DOES NOT SUPPORT SP)*/


        dbcmd.CommandText = sqlQuery;
        /*set the CommantText field to the above query(see doc for more details)*/

        IDataReader reader = dbcmd.ExecuteReader();
        /*Execute the above set query.
         This functions returns a value of the type IDataReader(see doc for more details).
         here is an example of how this works:
         consider the query used above(select id,username,password from user):
         id is of type int
         username and password are of type string
         IF the query returns 10 rows, the reader will also have 10 rows with
         id,username and password in each row in the same squence.
         */

        while (reader.Read())//(Reads one row(tupple) from the reader and returns true if read successfull(i.e
data available;
        {   //The next time the Read method is called, it reads the next row until no rows are avalable(see
documentation).

            /*Now to read data from the reader object, you use the Get( with appropriate type cast ) method.
             * In this case , each tuple contains id,username and passwword(in this order), So to get this you
call
             * the appropriate Get method with the proper index of the data.
             * i.e id will have index 0 and the type is int ,so we use reader.GetInt32(0);
             * username will have index 1 and the type is String(or Text),so we use reader.GetString(1);
             * and so on;
             */

            int id=reader.GetInt32(0);
            string username=reader.GetString(1);
            string password=reader.GetString(2);


            /*After retriving the data we can do the required operations like make changes to the UI,database
etc*/
            //EXAMPLE(code doesnt make any sense.just for demonstration)
            Text datatext = gameObject.GetComponent<Text>();
            if (password.Length > 8)
            {
                datatext.text = id +"  "+ username +"  "+ password;
            }
            else
            {
                datatext.text = "invalid password";
                datatext.color = Color.red;
            }
        }

        //After running all operations ,you can close the connection to the database;
        reader.Close();

        dbcmd.Dispose();

        dbconn.Close();

    }
}
```

DOCUMENTATION: https://www.mono-project.com/docs/database-access/providers/sql/

CONNECTING FRONT END(UNITY) WITH MY-SQL SERVER (supports SP)

```
using System;
using System.Data;
using System.Data.SqlClient;

public class Test
{



    public void AnyEVENTListner()
    {

        string connectionString =
            "Server=TestServer;" +
            "Database=test;" +
            "User ID=sa;" +
            "Password=MightyMighty;";


        IDbConnection dbcon;

        using (dbcon = new SqlConnection(connectionString)) {
            dbcon.Open();
            using (IDbCommand dbcmd = dbcon.CreateCommand()) {
                string sql =
                    "SELECT fname, lname " +
                    "FROM employee";
                dbcmd.CommandText = sql;
                using (IDataReader reader = dbcmd.ExecuteReader()) {
                    while(reader.Read()) {
                        string FirstName = (string) reader["fname"];
                        string LastName = (string) reader["lname"];
                        Console.WriteLine("Name: " +
                             FirstName + " " + LastName);
                    }
                }
            }
        }
    }
}
```

UI DESIGN


BROWSE
GAMING
MUSIC
TWITCH R
LOG IN
SIGNUP
MY SUBSCRIPTIONS
ABC
DEF
HIJ
KLM
NOP
QRS
TUV
XYZ
SEARCH
TITLE
DESCRIPTION:
YOU MAY ALSO LIKE
VIDEO 1
VIDEO 2
VIDEO 3
VIDEO4