UNIVERSITY OF CALGARY

CPSC 471 W2020

LEC 01 TUT 02

Project Final Report

Team Members
Ejaaz LAKHANI
30025543
Khoa NGUYEN
30024087
Yousaf RAJA
30027772

Instructor
Abdullah
Teaching Assistant
Tanuja SUDHAKAR

April 6, 2020

Abstract. Database Management Systems (DBMS), are a major component of computer science and data storage. Specifically, DBMS refers to the software that interacts with end users, and applications, as well as the database that is used to capture and store the data. We can define a database as being sets of related data and its organization, as well as its interaction with different components of related data. Some common examples can include spreadsheets (such as excel), or any system where information is placed, gathered and analyzed.

Databases are specifically used to model varieties of real world situations, including but not limited to hotel bookings, customer information storage for loyalty programs etc. As a matter of fact, DBMS are used almost everywhere there is some sort of client-administrator relationship.

We decided to conceptualize this by building an API that models an online shopping system. Our system focuses purely on the trading aspect of the idea, where users can create profiles for items they no longer need. They can then view items put up by other users, generating requests for items they want. Users interested in each others items will then be able to chat with each other to confirm or deny the trade.

1. Introduction

The problem with which we are concerned is to build a database API to model some real-world problem or situation. Our idea, is that the world of online trading has limitations in that there are no apps specifically focused around trading alone. So we sought to create a system that does this.

Users will be able to create different profiles of items they are no longer in need of. They can then browse through other items that are up for trade indicating items in which they are interested. If both parties are interested in each other's item(s) they can open a chat window where they can arrange specifics.

We consider this problem to be interesting because it involves a different perspective to the trading game. Our solution revolves around the idea of e-commerce. A lot of people have useless junk lying around, and as they say "One persons' trash is another persons' treasure." We believe everyone has certain items they don't want to sell and may be too expensive to buy. By establishing a trading perspective we can ensure that people can match with those who are exclusively interested in each other's items.

2. Problem Description

We as a group have chosen to create a database to model an online shopping/trading system where in which users can upload pictures of the items they are no longer in need of.

There do exist other websites and apps that incorporate the same idea as us. For example Kijiji and Mercari.

Kijiji - is a popular app that exists worldwide. People can make postings of the items they are no longer in need for and list it along with a price or a OBO (or best offer). People can filter items by category (item type) and distance from their current location.

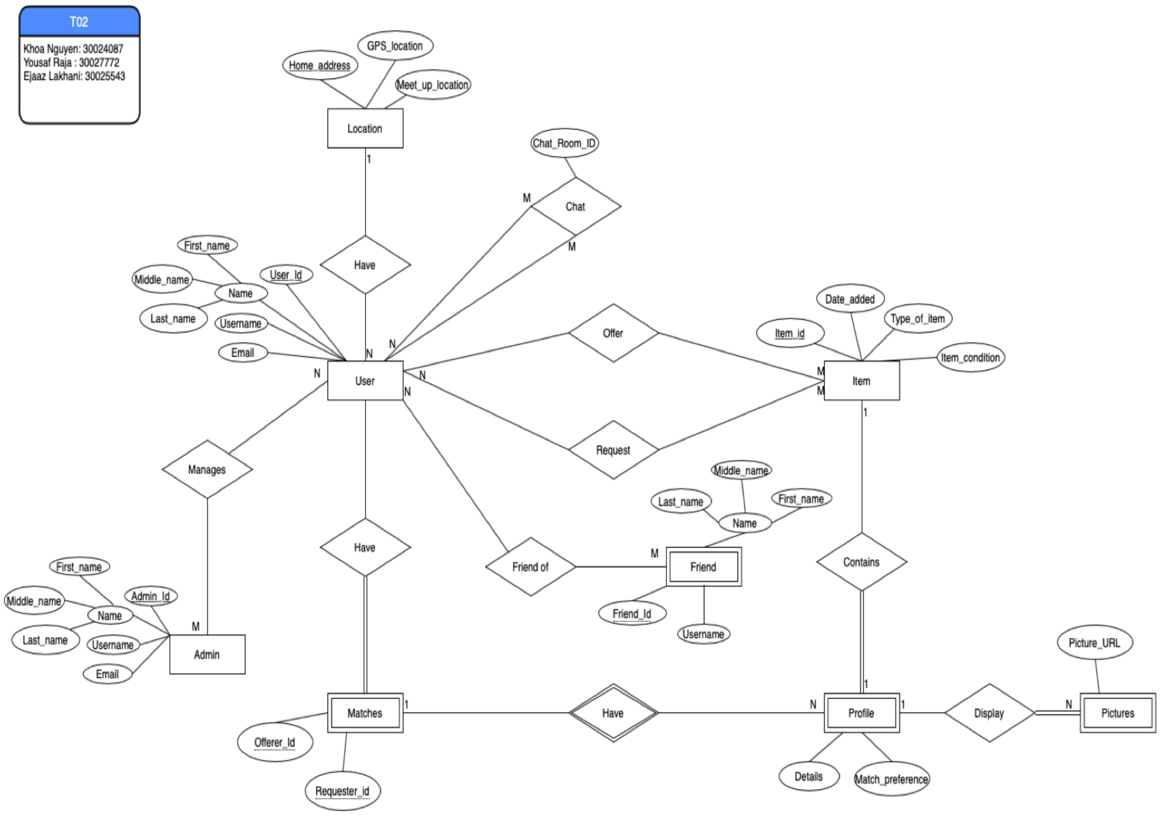
Mercari - is similar to kijiji in that users can make postings of their items along with sale price. Again you can use filters such as distance and item type.

Both of the previously mentioned apps are based on monetary transactions, with Kijiji having an option of swapping items instead. Kijiji states in 2018 that around 6% of Canadians trade via swapping rather than money, with the rest of the majority favouring monetary trades or donations. We hope to provide a platform where users are not swamped in a market favouring monetary gain and instead interact in a more communicational-trading system.[1] Mercari on the other hand, is almost purely monetary based transaction application, with the transactions generally favouring the buyers looking to make deals that benefit more towards the buyer side as well. With the proposed idea we hope to allow more favourable transactions for both sides whose makes exchanges based on each item value using means not just monetary. [2]

Knowing that other apps that exist that have similar ideas to ours are limited in they largely revolved around buying and selling. In the case they do offer trading they are limited in the case that they only offer trading for the same item (such as car for car) or they make each post public, in that someone can see you are offering up a car and can message you with any random trade. Our API ensures trading can be across various items and the only way messaging can be initiated is that both parties are interested in each other's items.

3. Project Design

Consider the following diagram.

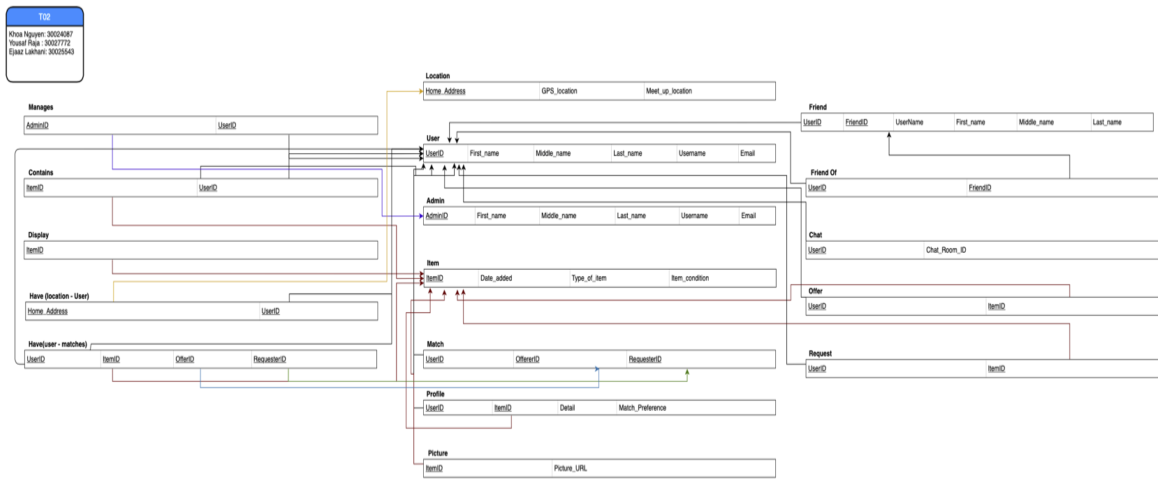


The above diagram is a complete entity relational diagram for our system. It depicts how the entities are related to each other and the individual attributes each entity possesses.

Our program contains an administrator. Each administrator has a unique admin ID and is additionally identified through their first middle and last name as well as their email and admin username. Administrators will be responsible for managing each user to ensure they are complying with the rules and regulations of the system.

Our program will have multiple users. Each user is identified uniquely through a user ID, additionally each users information will also contain their first, middle and last name as well as their email and chosen username. Additionally users can make friends with other users they frequently trade with. The users friend is another user containing the same attributes as user. Users (or friends) who express interest with each others items will be taken to a chat room, with each chat room having a unique room ID.
Each user has a location which will be uniquely identified by their home address as well as their GPS location and a meet up location for when a trade is established.
Users can offer up an item for trade or they can request another users item. Each item will contain an unique item ID as well as be described by date added the type of item (such as car, phone etc.) and the condition the item is in.
Items will be stored in profiles, users can create multiple profiles for numerous items. Profiles are identified through the profile details and the match preference for the profile. Moreover, each profile contains multiple pictures displaying the item in order for other users to see the items state and what it looks like so they can better establish interest. Finally user profiles and users themselves are related through an the Matches entity. Matches contain the IDs of both the offerer and requester of the profile storing the item.

4. Implementation



The above diagram depicts the relational model of our system based upon the Entity Relational Diagram shown in the previous section.

For our implementation, our database, was built in My-SQL workbench. We modelled our DBMS using python (under the file main.py) and created a series of API endpoints via postman.

Our program consisted of the following test SQL statements (which are also stored within the attatched file storedProcedures.dms)

(1)
```
CREATE DEFINER=‘admin‘@‘%‘ PROCEDURE ‘e1‘(IN UserID
INT)
BEGIN
SELECT   *
FROM   ‘471‘.user
WHERE   ‘471‘.user.User_Id=UserID;
END
```

(2)
```
CREATE DEFINER=‘admin‘@‘%‘ PROCEDURE ‘e2‘()
BEGIN
SELECT   Friend.First_name, Friend.Friend_Id, Friend.Username,
User.First_name, User.User_Id, User.Username

FROM   ‘471‘.Friend, ‘471‘.Friend_of, ‘471‘.User
WHERE   User.User_Id = Friend_of.User_Id AND Friend.Friend_Id
= Friend_of.Friend_Id;
END
```

(3)
```
CREATE DEFINER =‘admin‘@‘%‘ PROCEDURE ‘e3‘
(IN UserID INT, IN UserFNAME VARCHAR(255), IN UserMNAME
VARCHAR(255), IN UserLNAME VARCHAR(255), IN User-
NAME VARCHAR(255), IN UserEmail VARCHAR(255))

BEGIN
INSERT INTO   ‘471‘.User
VALUES   (UserID, UserFName, UserMName, UserLName,
UserName, UserEmail);
END
```

(4)
```
CREATE DEFINER =‘admin‘@‘%‘ PROCEDURE ‘e4‘
(IN UserID INT, IN UserLName VARCHAR(255), IN User-
Name VARCHAR(255))

BEGIN
UPDATE   ‘471‘.User
SET   Last_name = UserLName, Username = UserName
WHERE   User_Id = UserID;
```

END

(5) **CREATE DEFINER** =‘admin‘@‘%‘ PROCEDURE ‘e5‘()

BEGIN
SELECT U1.Username, Matches.Offerer_Id, U2.Username, Matches.Requester_Id
FROM ‘471‘.User as U1, ‘471‘.User as U2, ‘471‘.Matches
WHERE U1.User_Id = Matches.Offerer_Id **AND** U2.User_Id = Matches.Requester_Id;
END

(6) CREATE DEFINER=‘admin‘@‘%‘ PROCEDURE ‘e6‘()

BEGIN
SELECT U1.Username, Matches.Requester_Id, U2.Username, Matches.Offerer_Id
FROM ‘471‘.User as U1, ‘471‘.User as U2, ‘471‘.Matches
WHERE U1.User_Id = Matches.Requester_Id **AND** U2.User_Id = Matches.Offerer_Id;
END

(7) **CREATE DEFINER** =‘admin‘@‘%‘ PROCEDURE ‘e7‘(**IN** ItemID INT, **IN** DateAdded DATE, **IN** TYPEofItem VARCHAR(255), **IN** ItemCondition VARCHAR(255), **IN** UserID INT, **IN** DetailsText TEXT, **IN** MatchPreference TEXT)

BEGIN
INSERT INTO ‘471‘.Item
VALUES (ItemID, DateAdded, TYPEofItem, ItemCondition);
INSERT INTO ‘471‘.Profile
VALUES (UserID, ItemID, DetailsText, MatchPreference);
END

(8) **CREATE DEFINER** =‘admin‘@‘%‘ PROCEDURE ‘e8‘()

BEGIN
SELECT *
FROM ‘471‘.item;
END

(9)
```
CREATE DEFINER =‘admin‘@‘%‘ PROCEDURE ‘e9‘(IN
ItemID INT)

BEGIN
DELETE
FROM   ‘471‘.Item
WHERE   Item.Item_Id = ItemID;
END
```

(10)
```
CREATE DEFINER =‘admin‘@‘%‘ PROCEDURE ‘e10‘()

BEGIN SELECT   Item.Item_Id, User.User_Id
FROM   ‘471‘.User, ‘471‘.Profile, ‘471‘.Item
WHERE   User.User_Id = Profile.User_Id AND Profile.Item_Id
= Item.Item_Id;
END
```

(11)
```
CREATE DEFINER =‘admin‘@‘%‘ PROCEDURE ‘e11‘()

BEGIN
SELECT   Admin.Admin_Id, User.User_Id
FROM   ‘471‘.Admin, ‘471‘.User;
END
```

(12)
```
CREATE DEFINER =‘admin‘@‘%‘ PROCEDURE ‘e12‘(IN
ItemID INT, IN PictureURL VARCHAR(255))

BEGIN
INSERT INTO   ‘471‘.pictures
VALUES   (PictureURL, ItemID);
END
```

5. API Documentation

Our project API documentation can be found via the following link.

https://documenter.getpostman.com/view/11077819/Szf26WX2?version=latest

6. Conclusion

Throughout the course of the semester we worked on implementing our design or our proposed project. Our proposal was to implement an application that provides efficient trade experience that is unique from already existing e-commerce apps such as Kijiji or Mercari.
Through our work we designed ERD and Relational Models for our system. These diagrams aided us in creating functional endpoints for our system.

References

[1] https://www.kijiji.ca/kijijicentral/app/uploads/2016/08/Kijiji-Index-Report-2018_EN_Final_web-2.pdf https://www.mercari.com/

Existing application similar in theory, but different in terms of execution, purely money for item transactions

[2] https://www.ecommercebytes.com/2019/02/09/sellers-choice-2019-marketplace-ratings-mercari/