Report for Final Project template

Title of the Project: Second-hand transaction information retrieval platform

Name of the students: Haoyang Yuan, Ruiying Wang, Chen Liang

I. **Introduction**: We collected the data on second-hand merchandise from four second-hand trading websites and build a database to store it. Then we write an application and connect it with our database so that the user can easily search the information of merchandise by using this application.

Motivation: We choose this idea because there are many platforms on the Internet. It's not convenient that sometimes we want to search for the best merchandise with good price, good location, and more accurate searching results, then we need to search on many different platforms. But now we connect the four biggest second-hand transaction platforms for users to search and compare the results.

Describe the application: There are five options for users to choose from: item name, item (seller) location, item category, min price, and max price. Those options are all optional for users to choose when searching for. After a user completes his choice, click "Search", then the result of information on the merchandise will be shown in the same window.

Details: We set an error catch to remind users when they input the wrong information, such as inputting an alphabet in price blank. We also set a small function that a user must press "Enter" after each filling the blank to make sure the user searches for what he wants. After each time "Search" is clicked, all blank will be cleaned.

Describe the overall organization of the report and task assignment for each team member:

Code done by Haoyang with Ruiying Wang, Chen Liang Help.

Report done by Haoyang and Ruiying Wang help.
PowerPoint and Video done by Ruiying.

II. Our Implementation

2.1 Description of the system architecture

The system is built based on Java, using JDBC to connect to mysql database. The GUI user application is written in java.

2.2 Description of the dataset

Dataset is gotten from four websites by using a crawler.

There are four datasets: user, platform, item, and category

Properties of each dataset:

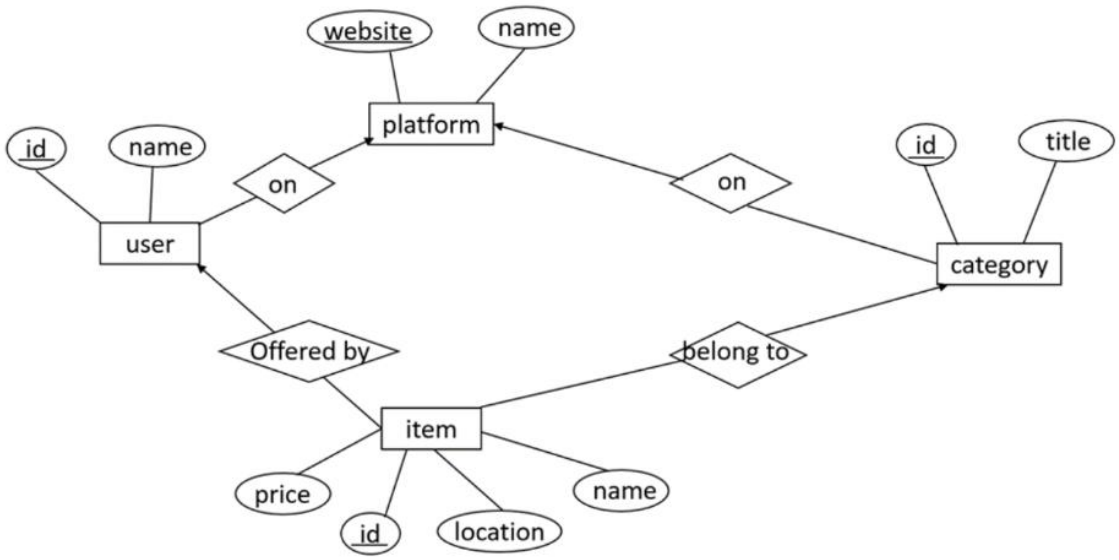
User: uid: user id, uname: user name, ulocation: user location, purl: website link, foreign key references Platform, primary key

Item: iid: item id, iname: item name, iprice: item price, cid: category id, uid: user id, two foreign keys and one primary key

Category: cid: category id, purl: platform link, ctitle, category name, one foreign key and one primary key

Platform: pname: platform name, purl: platform link, one primary key

2.3 ER diagram (final version from the previous checkpoint copied here)



We've revised the ER diagram, because we find that having an entity set for each category and platform is redundant representation. In addition, we have added ids to each user, category, and listed item, where those ids are website-assigned. These ids combined with platform forms a weak entity set.

2.4 Relational model (final version from the previous checkpoint copied here)

Relational schema:

a. platform(website, platform-name): both website and platform-name are keys, website is the primary key

b. user(user-id, user-name， platfrom): (user-id, platform) is the primary key

c. category(category-id, category-title， platfrom): (category-id, platform) is the primary key and the (category-title, platform) is another key
d. item(item-id, price, location, item-name, user, category): (item-id, user) is the primary key and the (item-id, category) is another key

Functional Dependencies

platform-name→website
website→platform-name
(platform, category-title)→(platform, category-id)
(platform, category-id)→(platform, category-title)
(platform, user)→platform
(platform, category)→platform
(platform, item)→(platform, user)
(platform, item)→(platform, category)

Simplified:

platformname→website
website→platformname
(platform, category-title)→category-id
(platform, category-id)→category-title
(platform, item)→user
(platform, item)→category

Platform, Category, User, and Item tables are 3NF. Everything on the right side is part of the candidate keys in the respective tables.

2.5 Implementation: description of the prototype

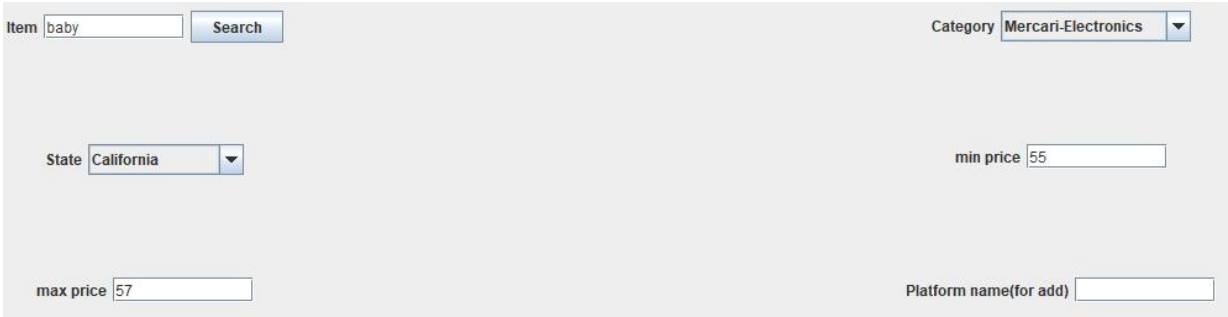
Firstly, get data from websites by a crawler. Then build a database to import all data. Then we create a user application and connect it to local database.

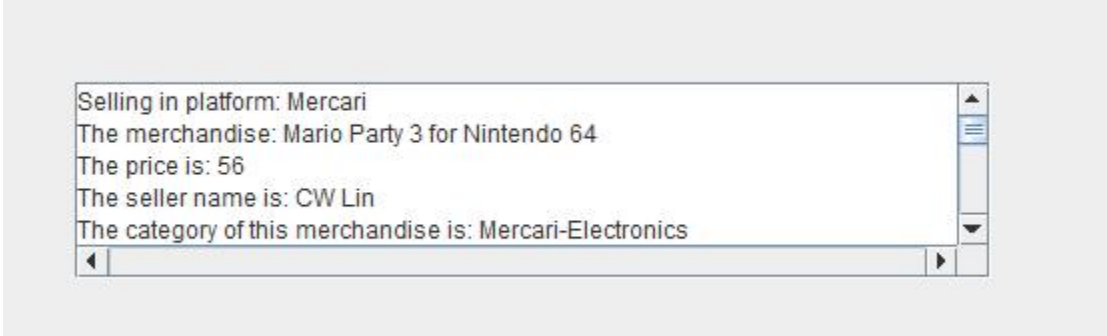
2.6 Evaluation: describe how you test your application (e.g, create testing scenarios or queries or something else, running your application through these scenarios/queries/etc.., checking the returned results and see how the results make sense or not).

Here we test five scenarios:

1. Query: find the items that the key word of the item is baby ,the category belongs to the Men, the location for the item is Wisconsin and the price is between 0 -100.
   SQL: select p.pname, i.iname, i.iprice, u.uname, c.ctitle, u.ulocation
   -> from Item i, Category c, Platform p, User u
   -> where c.cid = i.cid and i.uid = u.uid and u.purl = p.purl and i.iprice >= 0 and i.iprice <= 100 and u.ulocation = " Wisconsin" and i.iname like '%baby%' and c.ctitle = 'Men';

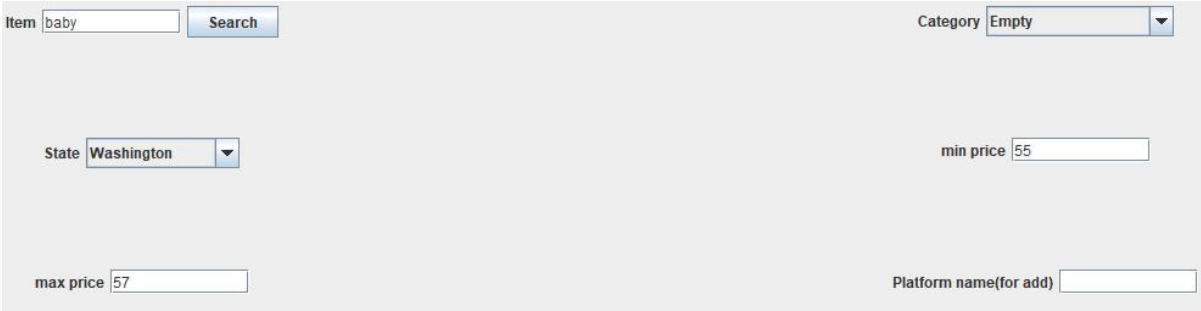
2. Query: find the name of items of which the price is between $55 to $57, the category belongs to elctronics and the state is California .
   SQL: select p.pname, i.iname, i.iprice, u.uname, c.ctitle, u.ulocation
   -> from Item i, Category c, Platform p, User u
   ->where c.cid = i.cid and i.uid = u.uid and u.purl = p.purl and iprice > 55 AND iprice < 57; u.ulocation = " California" and c.ctitle = "Electronics";



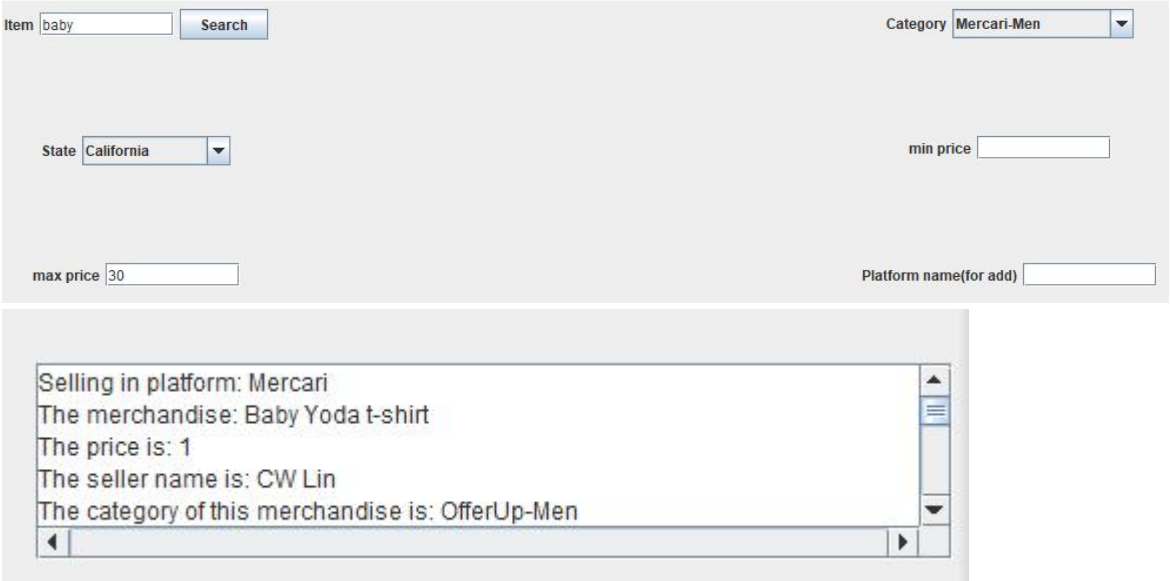


3. Query: find the items that the key word of the item is baby, the location for the item is Wisconsin and the price is between 55 -57.

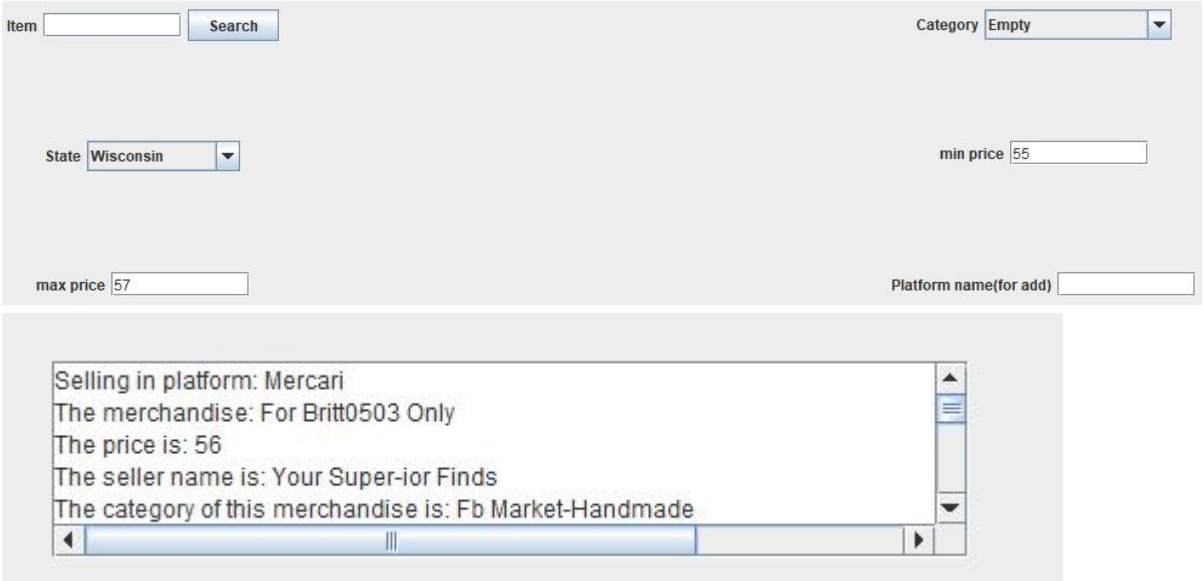
   SQL: select p.pname, i.iname, i.iprice, u.uname, c.ctitle, u.ulocation
   -> from Item i, Category c, Platform p, User u
   -> where c.cid = i.cid and i.uid = u.uid and u.purl = p.purl and i.iprice >= 55 and i.iprice <= 57 and u.ulocation = " Wisconsin" and i.iname like '%baby%' ;



4. Query: find the items that the key word of the item is baby ,the category belongs to the Men, the location for the item is California and the price is less than 30.
   SQL: select p.pname, i.iname, i.iprice, u.uname, c.ctitle, u.ulocation
   -> from Item i, Category c, Platform p, User u
   -> where c.cid = i.cid and i.uid = u.uid and u.purl = p.purl and i.iprice <= 30 and u.ulocation = " California" and i.iname like '%baby%' and c.ctitle = 'Men';



5. Query: find the items that the location for the item is Wisconsin and the price is between 55 -57.
   SQL: select p.pname, i.iname, i.iprice, u.uname, c.ctitle, u.ulocation
   -> from Item i, Category c, Platform p, User u
   -> where c.cid = i.cid and i.uid = u.uid and u.purl = p.purl and i.iprice >= 55 and i.iprice <= 57 and u.ulocation = " Wisconsin" ;



III. Storage Procedure

(1) create a stored procedure called category_List that selects all the categories from the category table.

Syntax:

```
delimiter //
create procedure category_List()
   -> begin
   -> select * from Category;
   -> end //
delimiter ;

call category_List();
```

Result:

cid	purl	ctitle
FBM01	https://www.facebook.com/marketplace/	Fb Market-Woman
FBM02	https://www.facebook.com/marketplace/	Fb Marketi-Men
FBM03	https://www.facebook.com/marketplace/	Fb Market-Kids
FBM04	https://www.facebook.com/marketplace/	Fb Market-Home
FBM05	https://www.facebook.com/marketplace/	Fb Market-Collectibles
FBM06	https://www.facebook.com/marketplace/	Fb Market-Beauty
FBM07	https://www.facebook.com/marketplace/	Fb Market-Electronics
FBM08	https://www.facebook.com/marketplace/	Fb Market-Outdoors
FBM09	https://www.facebook.com/marketplace/	Fb Market-Handmade
FBM10	https://www.facebook.com/marketplace/	Fb Market-Other
MCR01	https://www.mercari.com/us/category/1/	Mercari-Woman
MCR02	https://www.mercari.com/us/category/1/	Mercari-Men
MCR03	https://www.mercari.com/us/category/1/	Mercari-Kids
MCR04	https://www.mercari.com/us/category/1/	Mercari-Home
MCR05	https://www.mercari.com/us/category/1/	Mercari-Collectibles
MCR06	https://www.mercari.com/us/category/1/	Mercari-Beauty
MCR07	https://www.mercari.com/us/category/1/	Mercari-Electronics
MCR08	https://www.mercari.com/us/category/1/	Mercari-Outdoors
MCR09	https://www.mercari.com/us/category/1/	Mercari-Handmade
MCR10	https://www.mercari.com/us/category/1/	Mercari-Other
OFUP01	https://offerup.com/	OfferUp-Woman
OFUP02	https://offerup.com/	OfferUp-Men
OFUP03	https://offerup.com/	OfferUp-Kids
OFUP04	https://offerup.com/	OfferUp-Home
OFUP05	https://offerup.com/	OfferUp-Collectibles
OFUP06	https://offerup.com/	OfferUp-Beauty
OFUP07	https://offerup.com/	OfferUp-Electronics
OFUP08	https://offerup.com/	OfferUp-Outdoors
OFUP09	https://offerup.com/	OfferUp-Handmade
OFUP10	https://offerup.com/	OfferUp-Other

30 rows in set (0.00 sec)

(2) create a stored procedure selects all the platforms from the Platform table.

Syntax:

```
delimiter //
create procedure platform_list()
   -> begin
   -> select * from Platform;
   -> end //
delimiter ;

call platform_list();
```

Result:

```
+-----------------+-----------------------------------------+
| pname           | purl                                    |
+-----------------+-----------------------------------------+
| OfferUp         | https://offerup.com/                    |
| Facebook Market | https://www.facebook.com/marketplace/   |
| Mercari         | https://www.mercari.com/us/category/1/  |
+-----------------+-----------------------------------------+
3 rows in set (0.04 sec)

Query OK, 0 rows affected (0.04 sec)
```

(3) creates a stored procedure that finds all items which price equal $56

Syntax:

```
delimiter //
create procedure GetItemByPriceCost(in price_cost int)
   -> begin
   -> select * from Item where iprice = price_cost;
   -> end //
   delimiter ;

   call GetItemByPriceCost(56);
```

Result:

iid	iname	iprice	cid	uid
Item02273	John Galliano ombre with nice frames sunglasseswirh box cloth	56	FBM01	User-FB01090
Item02392	NWT New Large Women Ralph Lauren Brown Leather Tote Satchel Purse Handbag Feet	56	FBM01	User-FB01209
Item02811	Mens Lululemon Kahuna Shorts 36	56	OFUP01	User-OP00400
Item07207	Baby Boy Bundle	56	OFUP03	User-OP01627
Item07238	Ralph Lauren Polo Sweatsuit	56	OFUP03	User-OP01658
Item07432	Set of Four (4) New Pottery Barn Halloween Mugs 2 GHOST Figural 2 BOO	56	MCR04	User-MR03207
Item14268	Mario Party 3 for Nintendo 64	56	MCR07	User-MR05573
Item17089	Warrior custom Golf 2 Pc Wood set 5 and 7 in right Handed	56	MCR08	User-MR06908
Item19266	For Britt0503 Only	56	FBM09	User-FB06729

9 rows in set (0.35 sec)

IV. Interface:

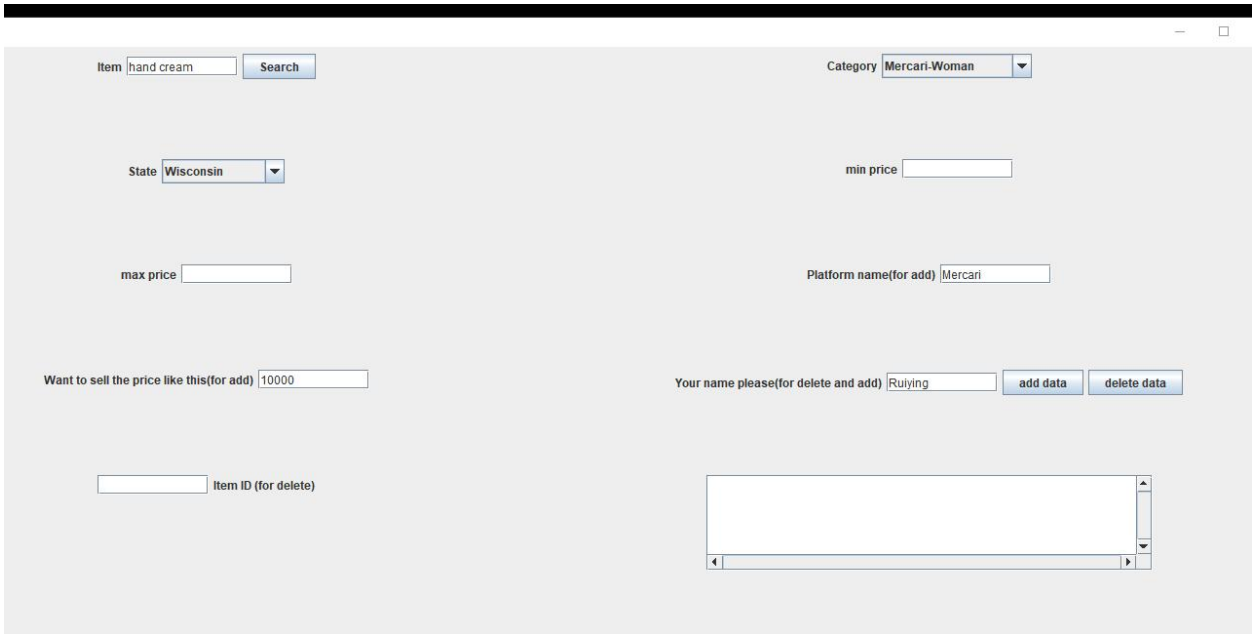
Insert new data into database:

Try to add a new data and the information is that a UW-madison student Peter Potter plans to sell his old-school t-shirt at the price of $ 78 in a new platform UW-SWAP. The Queries show below:

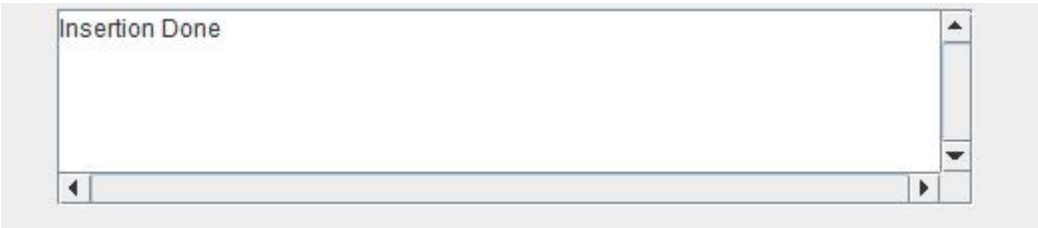
Insert data to Platform table	Syntax: insert into Platform -> Values('Mercari', 'https://www.mercari.com/us/category/1/');
Insert data to Category table	Syntax: insert into Category -> Values('MCR01', 'https://www.mercari.com/us/category/1/', 'Mercari-Woman');
Insert data to User table	Syntax: insert into User -> Values('User-MR08666','Ruiying', 'Wisconsin', 'https://www.mercari.com/us/category/1/');
Insert data to Item table	Syntax: insert into Item -> Values('Item23141','hand cream', 10000, 'UWS01','User-UWS00001');

When running on the application, it could also works successfully. The processes shows below:

(1) Set relevant information of the item to be added



(2) The application shows the item has bee added.



(3) Search the item in the database to check whether the item has added.

The result shows that the item has been inserted into the database.

Delete data from database:

Delete data we just added in the insert new data interface. The Queries show below:

delete data from User table	Syntax:
	set foreign_key_checks = 0;
	delete from User where uname = 'Ruiying';
	set foreign_key_checks = 1;

delete data from Item table	Syntax:
	set foreign_key_checks = 0;
	delete from Item where uid = 'User-MR08666';
	set foreign_key_checks = 1;

When running on the application, it could also works successfully. The processes shows below:

(1) Set delete command in the application.

(2) The application shows that the deletion is done.

(3) Search the item in the database to check whether the item has been deleted.

There is no merchandise matched, which means that the item has been deleted successfully in the database.

V. Evaluation

(i) the objectives of the evaluation:

The objectives for the application of the database is that is effectively and efficiently for users to handle searching, inserting, deleting, and modifying the data related to user, second-hand item, category and platform, as well as queries that involve getting inputs from users and retrieving information from multiple sources.

(ii) test cases:

a. insert at least 50 new data into the database.

b. delete at least 50 data from the database.

c. update 10 data in the database.

d. searching at least 100 SQL queries from the database.

e. clear all the data from the database

(iii) how we evaluate these test cases:

For the test cases from a to e, the evaluation is made by the the correctness rate of the results and time cost to finish all the tests. The evaluation shows that the application could successfully deal with searching, inserting, deleting, and modifying the data in the database with fast speed.

VI. Conclusion

What do you learn from this project (both interesting and uninteresting points)? Have you found any relevant database knowledge you have learned in this course helpful and have you encountered any database-relevant issues that have been discussed in this course?

We learned how to build a user application and how to connect to a local MySQL database, which is a very important step for this project. The most interesting part of this course is to draw

the ER diagram. It's because there are so many varieties that we can make our own diagram and there is no absolutely correct answer.

VII. References (if any)

https://www.facebook.com/marketplace/
https://www.mercari.com/us/category/1/
https://offerup.com/
https://swap.wisc.edu/