ADB GROUP-3

“UB AUCTION SALE SYSTEM”

**Team members:**

1. Srivatsan Ananthakrishnan – [srananth@my.bridgeport.edu](mailto:srananth@my.bridgeport.edu)
2. Vetrivelan Vajra Velu – [vvajarave@my.bridgeport.edu](mailto:vvajarave@my.bridgeport.edu)

**Project name:**

**UB Auction Sale System**

**Project Initiation:**

* Database server: “Oracle DBMS at UB”
* Type of Architecture: 3-tier architecture (client-server architecture)

\*Server – Apache Tomcat 8

\*Middle Ware – hibernate framework

\*Web Service – Restful Services

\*Client- html Web pages / Android Mobile App

* Programming language: Java

**Problem Statement:**

An application to manage University of Bridgeport On-line Auction Sale System. UB Employee’s or Students should be able to register and login into the website and check for auctions on cars and furniture and electronic gadgets etc. and bid them. Complete information of User’s is maintained in the system. Different types of User’s are present in the system. Based on user roles and permission user should be able to create, bid and generate reports accordingly. System should be enabled with feedback modules, and should be able to provide feedback on an Auction or the system. System should be enabled with Enquiry module which allows to enquire about bidding item to the seller. System should be also enabled with complaint or request module regarding any issues to Admin. Auction is classified into Category Based on the Type like Vehicles, Furniture, Electronics, Books, Tools, Clothing & Accessories and Others. At the time of user registration, system should take answers for pre-defined security question in the system and system should use this when validating user when forgot password feature is used.

**Realistic Queries:**

1. List of auctions based on auction status per month, weekly or daily basic.
2. List of inactive bidder and seller in a month or in a week.
3. Highest bidder in a month or in a year.
4. Highest no of bid’s in an auction.
5. Total sales happened in the system annually/monthly.
6. Marginal value between auction start price and bidding price.
7. Admin should be able to generate feedback report from Seller/Bidder based on auctions types.
8. Seller/Bidder can register in the system to participate in auction.
9. Generate report for registration between a given period.
10. Users can view their own completed auction, ongoing auctions and cancelled auction between a given period.
11. User can edit their profile.
12. Forgot Password, prompts user to answer security question and validate user and allows to change password.
13. Managing roles and permissions to manager, supervisor and normal user by Admin.
14. User viewing my auction list and sort the list based on price, date, status etc.
15. User updating their own profiles.

**Assumption about Database:**

**Entity**

USERTABLE, AUCTION, SECURITY\_QUESTION, FEEDBACK, MESSAGE, ROLE, PERMISSION, SHIPPINGSUMMARY, FAVORITES, IMAGES, ROLE\_PERMISSION, SECURITYANSWERS, BIDS\_ON, ROLE\_USER.

**Entity Attributes**

**USERTABLE**: - userid, address, adminFlag, bidderFlag, dateofbirth , email, name ,lname,

mname ,password ,phoneno ,sellerFlag, sex, ubid , userType username

**ROLE**: - roleid , role name

**PERMISSION**: - permissioned, Permission Code

**ROLE\_PERMISSION:-** roleid, permissioned

**ROLE\_USER:-** userid, roleid

**SECURITY QUESTIONS**: - question Code, question

**SECURITYANSWERS:-** question Code, answers, userid

**AUCTION**: - auctionid, startdate, enddate, auction Category Type (CAR, Furniture, Electronics, others) material, dimensions ,type, brand, model, miles, color, quoted price, bidding price, year, status, desc.

**SHIPPING** **SUMMARY**: auction\_bidderid, delivery method, payment method, delivery status, delivery address, auctionid, userid

**BIDS\_ON:-** auction\_bidder\_Id,bidding\_price, datetime, auctionid\_auctionid, userid\_userid

**MESSAGE**: - messageid, Type of Message- (enquire, complaint), messagebody

**FAVORITES**: - creationDate, favid,

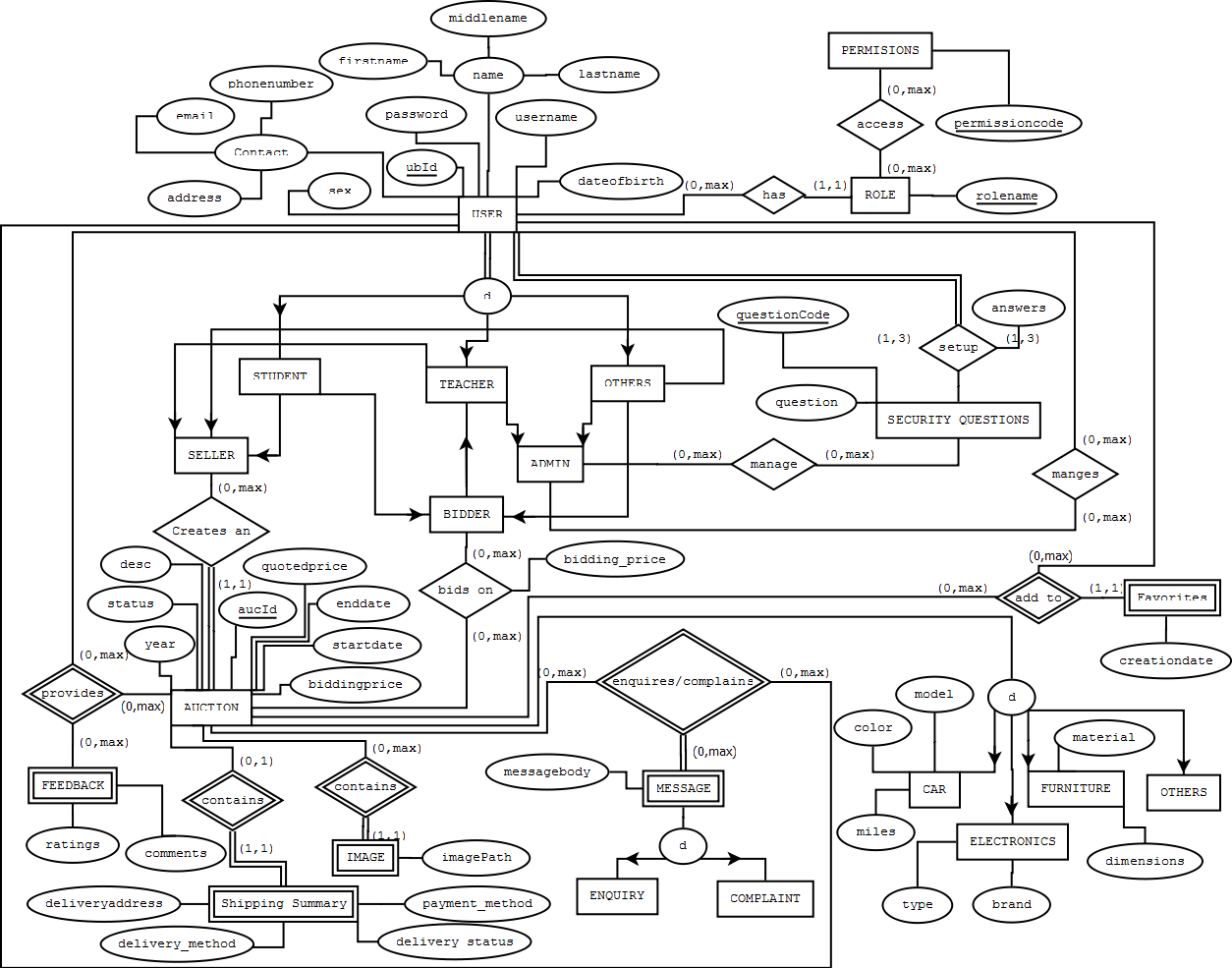
**FEEDBACK**: - ratings, comments, feedbackid

**IMAGE**: imagepath, imageid

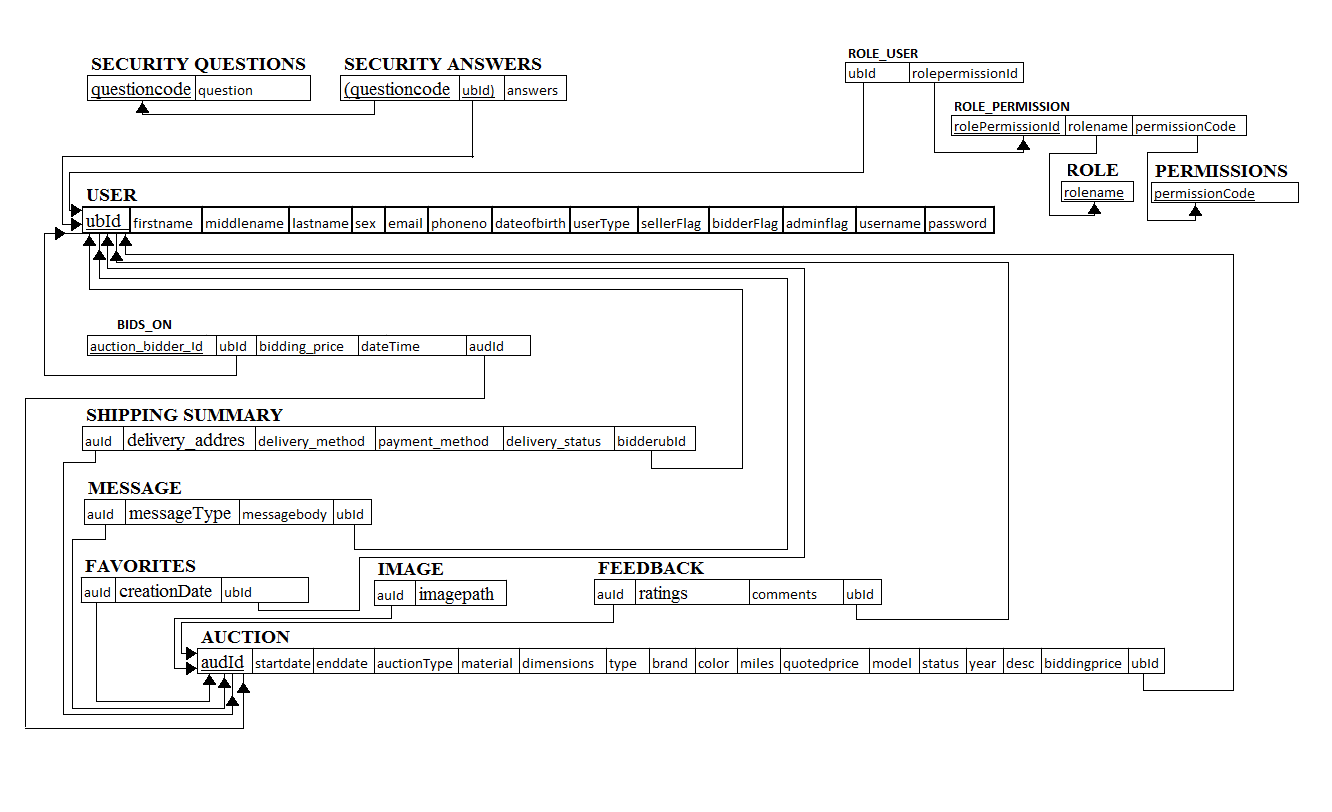
**Relationship between entities**

1. **USERTABLE** *has a* **ROLE**.
2. **ROLE** *have access to* **PERMISSION***.*
3. **USERTABLE (SELLER)** *Creates* an **AUCTION**.
4. **USERTABLE (BIDDER/SELLER)** *provide* **FEEDBACK** on **AUCTION**.
5. **AUCTION** *contains* **SHIPPINGSUMMARY**.
6. **AUCTION** *contains* **IMAGE.**
7. **USERTABLE (BIDDER/SELLER)** *enquires* on **AUCTION** with **MESSAGE**.
8. **USERTABLE** add **AUCTION** *to* **FAVORITES**.
9. **USERTABLE (ADMIN)** *Can* manage **USER**.
10. **USERTABLE** *setup answers to* **SECURITY QUESTION**.
11. **USERTABLE (ADMIN)** *mange(crud)* **SECURITY QUESTION**.

**Phase II: Enhanced Entity Relationship diagram:**



**Relational Schema diagram:**



**Relational Algebra for realistic quires**

1. SQL-select audId , avg(ratings) from feedback f group by audId;

RA:-Ans←audIdFavg(ratings), audId(FEEDBACK);

2.SQL:-select \* from USERS where creationDate between to\_date('15-JAN-17') and to\_date('17-MAR-17');

RA:-Ans← σ creationDate >=’15-JAN-17’ and creationDate <= ’17-MAR-17’(USERS);

3.select us.ubId, us.fname, us.lname, rp.permissionCode from USERS us JOIN ROLE\_USER ru ON ru.ubId = us.ubId JOIN ROLE\_PERMISSION rp ON rp.rolepermissionId = ru.rolepermissionId WHERE us.userType= 'Admin';

RA:-JOIN\_RESULT←(USERS rolepermissionId=rolePermissioId ⋈ ROLE\_PERMISSON)

RESULT←(σ userType=’Admin’ (JOIN\_RESULT)

Ans←π ubId,fname,lname,permissionCode(RESULT);

4.SQL:-select fname,ubId,lname from USERS where ubId in (select \* from (select bidderubId from BIDS\_ON where createdDateTime between to\_date('15-JAN-17') and to\_date('17-MAR-17') order by bidding\_price desc) where ROWNUM=1);

RA:-UBID\_MAXBID← rowNum=1F(π bidderubId (σ createdDateTime >=’15-JAN-17’ and createdDateTime <= ’17-MAR-17’(BIDS\_ON)));

JOIN\_RESULT←(USERS ubId=bidderubId ⋈ UBID\_MAXBID)

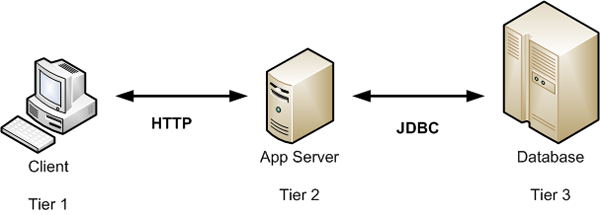
Ans←π ubId,fname,lname,permissionCode(JOIN\_RESULT);

5. SQL:-select au.audId, count(bo.audId) as cnt from BIDS\_ON bo, auction au where au.audId = bo.audId group by au.audId having count(\*) >2;

RA:-RESULT1←π audId (F count audid>2 BIDS\_ON)

Ans← π audId (RESULT1 audId= audId ⋈ AUCTION)

**System Diagram:**



The 3-tier architecture works as mentioned below.

1. Presentation logic - the user interface (UI) which displays data to the user and accepts input from the user. In our application it receives the request from mobile (Android) which is our client and the returns the response to the server which is Apache Tomcat server in our application.
2. Business logic - handles data validation, business rules and task-specific behavior.
3. Data Access logic - communicates with the database by constructing SQL queries and executing them via the relevant API.

**System Requirements:**

Server – Apache Tomcat 8

Platform – Java 1.8

Tools- Android Studio, Android SDK, Eclipse IDE,

Device / emulator

Functioning:

Compile and generate the war file and deploy the same using Eclipse.

Export dynamic web project as war file. Deploy the war file into Tomcat webabb folder.

To start Tomcat:

startup.bat

./ startup.sh

To run the application,

Step1. Build client application using Android Studio and export as .apk

Step2. Installing and handling an apk ,

* Copy project1.apk to Android device.
* Open File browser and Install project1.apk on device
* Click on project1 Icon from home screen

Step3. Run Project on Device/Emulators

**Conclusion**

The EER and relational schema for our “UB Auction Sale System” project has been implemented and the use of primary key and foreign keys from the above-mentioned relations are identified and implemented. We are implementing and demonstrating the functioning with Android GUI support as mentioned in the System Functioning section. We wish to deliver the expected output as per our provided requirements and specifications from Phase III and realistic queries.