**CS331 Project Phase 3 Report**

Team HaramBase

Shilei Lin, Joshua Folkerds, Matthew Kouniyom



*This project is dedicated to our hero, Harambe, the Gorilla that was.*

**Table of Contents**

[**PART I: Updated Diagrams** 3](#_Toc466992705)

[**PART II: Physical Database Design** 4](#_Toc466992706)

[**Part III: System Functionality Descriptions** 7](#_Toc466992707)

[Section One: Admin Subsystem: 7](#_Toc466992708)

[Section Two: Customer Subsystem 9](#_Toc466992709)

[**PART IV: SQL Code Components** 15](#_Toc466992710)

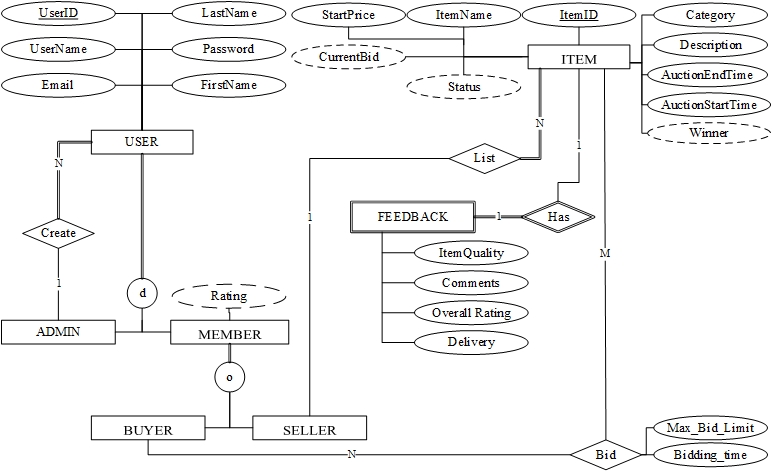
[**PART V: Highlighted Problems** 25](#_Toc466992711)

[**PART VI: Task Decomposition** 25](#_Toc466992712)

[**PART VII: Meeting Minutes** 26](#_Toc466992713)

# **PART I: Updated Diagrams**

1. **EER diagram:**



1. **Relational schema diagram**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ADMIN |  |  |  |  |  |  |  |
| UserID | UserName | Email |  | FirstName | LastName | Password | CreatorID |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Member |  |  |  |  |  | |  |  | | | |  |  |  |  |
| UserID | UserName | Email |  | FirstName | LastName | Password | CreatorID | | IsBuyer | IsSeller |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feedback |  |  | |  | |  | |  |
| ItemID | OverallRating | | ItemQuality | | Comments | | Delivery |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Item | | | | | |  |  |
| ItemID | ItemName | Category | StartPrice | Description | UserID | AuctionStartTime | AuctionEndTime |

|  |  |  |  |
| --- | --- | --- | --- |
| Bid | | | |
| UserID | ItemID | BiddingTime | MaxBidLimit |

# **PART II: Physical Database Design**

*The code is used for creating the databases and with comments on the left side*

|  |  |
| --- | --- |
| **Comments** | **DDL statements** |
| 1. This will create the ADMIN table of database, which has USERID as primary Key and other 5 attributes.   UNAME is another unique attribute. | DROP TABLE HARAMBASE\_ADMIN CASCADE CONSTRAINTS;  CREATE TABLE HARAMBASE\_ADMIN(  USERID INTEGER PRIMARY KEY,  UNAME VARCHAR(10) UNIQUE,  EMAIL VARCHAR(20),  FNAME VARCHAR(15),  LNAME VARCHAR(15),  PASSWORD VARCHAR(16),  CREATORID INTEGER,  FOREIGN KEY (CREATORID) REFERENCES HARAMBASE\_ADMIN(USERID)  ON DELETE SET NULL); |
| 1. This will create the MEMBER table of database, which has USERID as primary Key and other 8 attributes. UNAME is another unique attribute. | DROP TABLE HARAMBASE\_MEMBER CASCADE CONSTRAINTS;  CREATE TABLE HARAMBASE\_MEMBER(  USERID INTEGER PRIMARY KEY,  UNAME VARCHAR(10) UNIQUE,  EMAIL VARCHAR(20),  FNAME VARCHAR(15),  LNAME VARCHAR(15),  PASSWORD VARCHAR(16),  CREATORID INTEGER,  ISBUYER NUMBER(1, 0),  ISELLER NUMBER(1, 0),  FOREIGN KEY (CREATORID) REFERENCES HARAMBASE\_ADMIN(USERID)  ON DELETE SET NULL); |
| 1. This will create the ITEM table of database, which has ITEMID as primary Key and other 7 attributes. | DROP TABLE HARAMBASE\_ITEM CASCADE CONSTRAINTS;  CREATE TABLE HARAMBASE\_ITEM(  ITEMID INTEGER PRIMARY KEY,  ITEMNAME VARCHAR(150),  ITEMCATEGORY VARCHAR(50),  STARTPRICE NUMBER(6,2),  DESCRIPTION VARCHAR(300),  SELLERID INTEGER,  AUCTIONSTARTTIME TIMESTAMP(3),  AUCTIONENDTIME TIMESTAMP(3),  FOREIGN KEY (SELLERID) REFERENCES HARAMBASE\_MEMBER(USERID)  ON DELETE SET NULL); |
| 1. This will create the FEEDBACK table of database, which has ITEMID as primary Key and other 7 attributes. | DROP TABLE HARAMBASE\_FEEDBACK CASCADE CONSTRAINTS;  CREATE TABLE HARAMBASE\_FEEDBACK(  ITEMID INTEGER PRIMARY KEY,  OVERALLRATING NUMBER(3,1),  ITEMQUALITY NUMBER(2,1),  DELIVERY NUMBER(2,1),  COMMENTS VARCHAR(300),  FOREIGN KEY (ITEMID) REFERENCES HARAMBASE\_ITEM(ITEMID)  ON DELETE SET NULL); |
| 1. This will create the BID table of database, which has the combination of ITEMID and USERID as primary Key and other 2 attributes. | DROP TABLE HARAMBASE\_BID CASCADE CONSTRAINTS;  CREATE TABLE HARAMBASE\_BID(  USERID INTEGER,  ITEMID INTEGER,  BIDDINGTIME TIMESTAMP(3),  MAXBIDLIMIT NUMBER(6,2),  PRIMARY KEY (USERID, ITEMID),--THIS IS HOW YOU DEFINE PRIMARY KEY  FOREIGN KEY (USERID) REFERENCES HARAMBASE\_MEMBER(USERID)  ON DELETE SET NULL,  FOREIGN KEY (ITEMID) REFERENCES HARAMBASE\_ITEM(ITEMID)  ON DELETE SET NULL); |
| 1. This part of codes will insert users into the database. | INSERT INTO HARAMBASE\_ADMIN VALUES (0, 'admin', 'admin@harambase.org', 'ad', 'min', 'admin', NULL);  INSERT INTO HARAMBASE\_MEMBER VALUES (1, 'jfolks', 'jfolks@harambase.org', 'Josh', 'Folkerds', 'Folkerds', 0, 1, 1);  INSERT INTO HARAMBASE\_MEMBER VALUES (2, 'mkounn', 'mkounn@harambase.org', 'Matthew', 'Kounniyom', 'Kounniyom', 0, 1, 1);  INSERT INTO HARAMBASE\_MEMBER VALUES (3, 'slin', 'slin@harambase.org', 'Shilei', 'Lin', 'Lin', 0, 1, 1);  INSERT INTO HARAMBASE\_MEMBER VALUES (4, 'irahal', 'irahal@harambase.org', 'Imad', 'Rahal', 'Rahal', 0, 1, 1); |
| 1. This part of codes will insert items into the database. | INSERT INTO HARAMBASE\_ITEM VALUES (100, 'SJU Snapback Hat', 'Outdoor Gear',30,'For the Johnniest of Johnnies!', 1, TO\_TIMESTAMP('03-NOV-2016 11:00:00', 'DD-MM-YYYY HH:MI:SS'), TO\_TIMESTAMP('13-NOV-2016 12:00:00', 'DD-MM-YYYY HH:MI:SS'));  INSERT INTO HARAMBASE\_ITEM VALUES (101, 'DATABASES BOOK', 'BOOK',100, 'BRAND NEW', 3, TO\_TIMESTAMP('03-NOV-2016 11:00:00', 'DD-MM-YYYY HH:MI:SS'), TO\_TIMESTAMP('12-NOV-2016 12:00:00', 'DD-MM-YYYY HH:MI:SS'));  INSERT INTO HARAMBASE\_ITEM VALUES (102, 'CHINSES INSTANT NODDLES', 'FOOD', 20, 'COOK IN 5 MINS', 2, TO\_TIMESTAMP('03-NOV-2016 11:00:00','DD-MM-YYYY HH:MI:SS'), TO\_TIMESTAMP('10-NOV-2016 12:00:00','DD-MM-YYYY HH:MI:SS'));  INSERT INTO HARAMBASE\_ITEM VALUES (103, 'TACO', 'FOOD', 5, 'FRESH', 3, TO\_TIMESTAMP('03-NOV-2016 11:00:00','DD-MM-YYYY HH:MI:SS'), TO\_TIMESTAMP('10-NOV-2016 12:00:00','DD-MM-YYYY HH:MI:SS'));  INSERT INTO HARAMBASE\_ITEM VALUES (104, 'Abstarct Mathematics', 'BOOK', 20, 'BRAND NEW', 3, TO\_TIMESTAMP('03-NOV-2016 11:00:00'), TO\_TIMESTAMP('10-NOV-2016,12:00:00','DD-MM-YYYY HH:MI:SS'));  INSERT INTO HARAMBASE\_ITEM VALUES (105, 'INTRODUCTION TO Mathematics', 'BOOK', 40, 'BRAND NEW', 3, TO\_TIMESTAMP('03-NOV-2017 11:00:00','DD-MM-YYYY HH:MI:SS'), TO\_TIMESTAMP('10-NOV-2017 12:00:00','DD-MM-YYYY HH:MI:SS'));  INSERT INTO HARAMBASE\_ITEM VALUES (106, 'INTRODUCTION TO Mathematics', 'BOOK', 50, 'BRAND NEW', 3, TO\_TIMESTAMP('03-NOV-2016 11:00:00','DD-MM-YYYY HH:MI:SS'), TO\_TIMESTAMP('10-NOV-2017 12:00:00','DD-MM-YYYY HH:MI:SS')); |
| 1. This part of codes will insert feedback into the database. | INSERT INTO HARAMBASE\_FEEDBACK VALUES (100, 9.5, 3.0, 1.0,'The hat did not arrive at all, I recieved a notifcation of shipping from the seller. But got nothing.');  INSERT INTO HARAMBASE\_FEEDBACK VALUES (101, 8.0, 5.0, 4.0,'The BOOK IS GOOD.');  INSERT INTO HARAMBASE\_FEEDBACK VALUES (104, 5.0, 5.0, 4.0,'The NOODLE IS GOOD.');  INSERT INTO HARAMBASE\_FEEDBACK VALUES (102, 10.0, 5.0, 4.0,'The BOOK IS GOOD.'); |
| 1. This part of codes will insert bidding into the database. | INSERT INTO HARAMBASE\_BID VALUES(2,100,TO\_TIMESTAMP('04-NOV-2016 11:00:00','DD-MM-YYYY HH:MI:SS'),50);  INSERT INTO HARAMBASE\_BID VALUES(3,100,TO\_TIMESTAMP('04-NOV-2016 12:00:00','DD-MM-YYYY HH:MI:SS'),60);  INSERT INTO HARAMBASE\_BID VALUES(2,101,TO\_TIMESTAMP('04-NOV-2016 10:00:00','DD-MM-YYYY HH:MI:SS'),150);  INSERT INTO HARAMBASE\_BID VALUES(3,101,TO\_TIMESTAMP('05-NOV-2016 11:00:00','DD-MM-YYYY HH:MI:SS'),160);  INSERT INTO HARAMBASE\_BID VALUES(4,101,TO\_TIMESTAMP('06-NOV-2016 12:00:00','DD-MM-YYYY HH:MI:SS'),170);  INSERT INTO HARAMBASE\_BID VALUES(3,102,TO\_TIMESTAMP('06-NOV-2016 11:00:00','DD-MM-YYYY HH:MI:SS'),6);  INSERT INTO HARAMBASE\_BID VALUES(4,102,TO\_TIMESTAMP('05-NOV-2016 12:00:00','DD-MM-YYYY HH:MI:SS'),7);  INSERT INTO HARAMBASE\_BID VALUES(4,104,TO\_TIMESTAMP('05-NOV-2016 12:00:00','DD-MM-YYYY HH:MI:SS'),50);  INSERT INTO HARAMBASE\_BID VALUES(1,103,TO\_TIMESTAMP('11-NOV-2016 12:00:00','DD-MM-YYYY HH:MI:SS'),10);  INSERT INTO HARAMBASE\_BID VALUES(2,103,TO\_TIMESTAMP('11-NOV-2016 12:00:01','DD-MM-YYYY HH:MI:SS'),10); |

# **Part III: System Functionality Descriptions**

* We did not include the procedures, functions, views and triggers codes in this part because it will be duplicating with PART IV: SQL Code Components.
* However, we will still mention them in the verbose description.
* Some sample outputs have been reformatted for display only.

## Section One: Admin Subsystem:

**Functionalities for administrators (total of 5 functionalities)**

|  |
| --- |
| NO.1 Functionality: Admin login |
| The SQL calls functions ADMIN\_LOG\_IN\_FUNC with two input parameters userID and password. The function will return 1 or 0 as success or fail to login |
| SQL Queries: |
| SELECT ADMIN\_LOG\_IN\_FUNC('admin','admin') FROM DUAL; |
| Sample Output: |
| --EXPECTED: 1  -- ADMIN\_LOG\_IN\_FUNC('ADMIN','ADMIN')  -----------------------------------------  -- 1 |

|  |
| --- |
| NO.2 Functionality: View Users |
| The Admin will view all the users (non-admin) |
| SQL Queries: |
| SELECT USERID, UNAME, FNAME, LNAME, EMAIL, PASSWORD FROM HARAMBASE\_MEMBER; |
| Sample Output: |
| -- USERID UNAME FNAME LNAME EMAIL PASSWORD  ------------ ---------- --------------- --------------- -------------------- ----------  -- 1 jfolks Josh Folkerds jfolks@harambase.org Folkerds  --  -- 2 mkounn Matthew Kounniyom mkounn@harambase.org Kounniyom  --  -- 3 slin Shilei Lin slin@harambase.org Lin  --  -- 4 irahal Imad Rahal irahal@harambase.org Rahal |

|  |
| --- |
| NO.3 Functionality: Add Users |
| The Admin will can add a new user by the procedure. Also, by calling this procedure, we do not need to specify the actual userID because a trigger will generate it before insert into the database. The default ‘-1’ there will not be inserted into the database, but we still need one to pass as parameters. |
| SQL Queries: |
| Set serveroutput on;  DECLARE  STATUS INT;  BEGIN  ADD\_USER\_PRO(-1, 'hrambe', 'hrambe@harambase.org', 'Harambe', 'Gorilla', 'Gorilla', 0, 0, 0, STATUS);  DBMS\_OUTPUT.PUT\_LINE(STATUS);  ADD\_USER\_PRO(-1, 'lmatt', 'Matt@harambase.org', 'Matthew', 'Lynch', 'Lynch', 0, 1, 1,STATUS);  DBMS\_OUTPUT.PUT\_LINE(STATUS);  END; |
| Sample Output: |
| -- USERID UNAME FNAME LNAME EMAIL PASSWORD  ------------ ---------- --------------- --------------- -------------------- ----------  -- 1 jfolks Josh Folkerds jfolks@harambase.org Folkerds  --  -- 2 mkounn Matthew Kounniyom mkounn@harambase.org Kounniyom  --  -- 3 slin Shilei Lin slin@harambase.org Lin  --  -- 4 irahal Imad Rahal irahal@harambase.org Rahal  --  -- 5 hrambe Harambe Gorilla hrambe@harambase.org Gorilla  --  -- 6 lmatt Matthew Lynch Matt@harambase.org Lynch |

|  |
| --- |
| NO.4 Functionality: Sales Summary Report |
| Admin would like to view a sales summary, which is grouped by item category and then item id and shows the list of items sold with item info and the commission free from the sale. This will select information from SALES\_SUMMARY\_REPORT view |
| SQL Queries: |
| SELECT \* FROM SALES\_SUMMARY\_REPORT; |
| Sample Output: |
| --CATE ID NAME FINAL\_PRICE COMMISSION  ------------ ---------- ----------------------- --------------- --------------------  --BOOK 104 Abstarct Mathematics 21 1.05  --BOOK 101 DATABASES BOOK 151 7.55  --FOOD 102 CHINSES INSTANT NODDLES 7 0.35  --FOOD 103 TACO 11 0.55  --Outdoor Gear 100 SJU Snapback Hat 51 2.55 |

|  |
| --- |
| NO.5 Functionality: Sales Summary Report |
| Admin would like to view a list of sellers who sold items. It is grouped by user id and shows user info with commission fees each has paid with total income at the bottom. This will select information from SALES\_SUMMARY\_REPORT view |
| SQL Queries: |
| SELECT \* FROM OVERALL\_COMMISSION\_VIEW; |
| Sample Output: |
| --USERID USER\_NAME FIRST\_NAME LAST\_NAME EMAIL SELLER\_RATING COMMISSIONS  ------------ ---------- --------------- --------------- -------------------------------  -- 1 jfolks Josh Folkerds jfolks@harambase.org 9.5 2.55  -- 2 mkounn Matthew Kounniyom mkounn@harambase.org 10 .35  -- 3 slin Shilei Lin slin@harambase.org 7.7E+00 9.15 |

## Section Two: Customer Subsystem

**Functionalities for Customers (total of 13 functionalities)**

|  |
| --- |
| NO.1 Functionality: User Login |
| The SQL calls functions ADMIN\_LOG\_IN\_FUNC with two input parameters userID and password. The function will return 1 or 0 as success or fail to login |
| SQL Queries: |
| SELECT MEMBER\_LOG\_IN\_FUNC('slin','Lin') FROM DUAL; |
| Sample Output: |
| --EXPECTED: 1  -- MEMBER\_LOG\_IN\_FUNC('SLIN','LIN')  -----------------------------------------  -- 1 |

|  |
| --- |
| NO.2 Functionality: Update Profile |
| The SQL will invoke the procedure for updating properses. The procedure will return SUCC as indicator whether the update success or not. Different SUCC number means different failure condition User wishes to update their profile information. (For Seller and Buyer) |
| SQL Queries: |
| set serveroutput on;  declare  SUCC Integer;  begin  UPDATE\_PROFILE\_PRO(3,NULL, '2@A.COM',NULL,'BOBARINO',NULL,NULL,NULL,SUCC);  DBMS\_OUTPUT.PUT\_LINE(SUCC);  end;  SELECT \* FROM HARAMBASE\_MEMBER WHERE USERID = 2; |
| Sample Output: |
| --USERID UNAME EMAIL FNAME LNAME PASSWORD CREATORID ISBUYER ISELLER  ------------------ ---------- ---------- ---------- -------- -------- -----------  --2 mkounn [2@A.COM](mailto:2@A.COM) Matthew BOBARINO Kounniyom 0 1 1 |

|  |
| --- |
| NO.3 Functionality: Show List of Items |
| Seller wishes to see all the items they are listing (For Seller) |
| SQL Queries: |
| SELECT ITEM.ITEMID, ITEM.ITEMNAME, ITEM.ITEMCATEGORY, ITEM.AUCTIONSTARTTIME,  ITEM.AUCTIONENDTIME, HARAMBASE\_GETPRICE\_FUNC(ITEM.ITEMID) AS CURRENT\_BID,  GET\_STATUS\_FUNC(ITEM.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM ITEM  WHERE ITEM.SELLERID IN (SELECT M.USERID  FROM HARAMBASE\_MEMBER M)  AND ITEM.SELLERID = '3'  ORDER BY GET\_STATUS\_FUNC(ITEM.ITEMID) DESC, ITEM.ITEMID; |
| Sample Output: |
| --ITEMID ITEMNAME CATE AUCTION START TIME  AUCTION END TIME CURRENTBID STATUS  ---------------------------------------------------------------------------------------  --101 DATABASES BOOK BOOK 03-NOV-16 11.00.00.000000000 AM  12-NOV-16 12.00.00.000000000 PM 151 1  --103 TACO FOOD 03-NOV-16 11.00.00.000000000 AM  10-NOV-16 12.00.00.000000000 PM 11 1  --104 Abstarct Mathematics BOOK 03-NOV-16 11.00.00.000000000 AM  10-NOV-16 12.00.00.000000000 PM 21 1  --106 INTRODUCTION TO Mathematics BOOK 03-NOV-16 11.00.00.000000000 AM  10-NOV-17 12.00.00.000000000 PM 0 0  --105 INTRODUCTION TO Mathematics BOOK 03-NOV-17 11.00.00.000000000 AM  10-NOV-17 12.00.00.000000000 PM 0 -1  --------------------------------------------------------------------------------------- |

|  |
| --- |
| NO.4 Functionality: Show List of Bidders |
| Seller wishes to see all the bidders by his/her current item. (For Seller) |
| SQL Queries: |
| SELECT HB.BIDDINGTIME ,HM.UNAME AS USERNAME, HB.MAXBIDLIMIT AS MAX\_BID\_LIMIT  FROM HARAMBASE\_BID HB, HARAMBASE\_MEMBER HM  WHERE HB.USERID IN (SELECT HM.USERID  FROM HARAMBASE\_ITEM HM)  AND HB.ITEMID = 101  ORDER BY HB.BIDDINGTIME; |
| Sample Output: |
| --BIDDINGTIME USERNAME MAX\_BID\_LIMIT  --------------------------------- ---------- -------------  --04-NOV-16 10.00.00.000000000 AM mkounn 150  --05-NOV-16 11.00.00.000000000 AM slin 160  --06-NOV-16 12.00.00.000000000 PM irahal 170 |

|  |
| --- |
| NO.5 Functionality: Show Item Information |
| User wishes to view detailed information about an item they listed above. (For Seller) |
| SQL Queries: |
| SELECT ITEM.ITEMID, ITEM.ITEMNAME, ITEM.ITEMCATEGORY, ITEM.AUCTIONSTARTTIME,  ITEM.AUCTIONENDTIME, ITEM.DESCRIPTION  FROM HARAMBASE\_ITEM ITEM  WHERE ITEM.ITEMID = '101'; |
| Sample Output: |
| --ITEMID ITEMNAME CATE AUCTION START TIME  AUCTION END TIME DESCRIPTION  -----------------------------------------------------------------------------------------101 DATABASES BOOK BOOK 03-NOV-16 11.00.00.000000000 AM 12-NOV-16 12.00.00.000000000 PM BRAND NEW |

|  |
| --- |
| NO.6 Functionality: Add Item |
| Invoke Trigger, User wishes to add a new item to sell.. |
| SQL Queries: |
| SELECT \* FROM Harambase\_ITEM;  EXEC ADD\_ITEM\_PRO(1, 'TESTING', 'BOOK', 50, 'BRAND NEW', 3, TO\_TIMESTAMP('03-NOV-2016 11:00:00', 'DD-MM-YYYY HH:MI:SS'), TO\_TIMESTAMP('10-NOV-2017 12:00:00', 'DD-MM-YYYY HH:MI:SS'));  SELECT \* FROM Harambase\_ITEM; |
| Sample Output: |
| --ITEMID ITEMNAME CATE AUCTION START TIME  AUCTION END TIME DESCRIPTION  -----------------------------------------------------------------------------------------101 DATABASES BOOK BOOK 03-NOV-16 11.00.00.000000000 AM 12-NOV-16 12.00.00.000000000 PM BRAND NEW |

|  |
| --- |
| NO.8 and 9 Functionality: Search |
| The procedure will accept all the inputs and return a table. User wishes to search for an item to bid on. |
| SQL Queries: |
| set serveroutput on;  DECLARE  TID NUMBER;  KEYWORD VARCHAR2(200);  TCATEGORY VARCHAR2(200);  TAUCTIONSTARTTIME TIMESTAMP;  TAUCTIONENDTIME TIMESTAMP;  CURBIDMIN NUMBER;  CURBIDMAX NUMBER;  CURBID NUMBER;  STATUS NUMBER;  RESULTSET SYS\_REFCURSOR;  BEGIN  TID := NULL;  KEYWORD := 'DATA';  TCATEGORY := NULL;  TAUCTIONSTARTTIME := TO\_TIMESTAMP('01-NOV-2016 10:00:00','DD-MM-YYYY HH:MI:SS');  TAUCTIONENDTIME := TO\_TIMESTAMP('20-NOV-2017 11:00:00','DD-MM-YYYY HH:MI:SS');  CURBIDMIN := NULL;  CURBIDMAX := NULL;  CURBID := 0;  STATUS := 0;  SEARCH\_ITEM\_PRO-- TID,  KEYWORD,  TCATEGORY,  TAUCTIONSTARTTIME,  TAUCTIONENDTIME,  CURBIDMIN,  CURBIDMAX,  RESULTSET  );  DBMS\_OUTPUT.PUT\_LINE('ID | NAME | CATE | AUCTION STAR TIME | AUCTION END TIME| CURRENT BID | STATUS');  LOOP  FETCH RESULTSET  INTO TID, KEYWORD,TCATEGORY,TAUCTIONSTARTTIME,TAUCTIONENDTIME,CURBID,STATUS;  EXIT WHEN RESULTSET%NOTFOUND;    DBMS\_OUTPUT.PUT\_LINE(TID || ' | ' || KEYWORD || ' | ' || TCATEGORY|| ' | ' ||TAUCTIONSTARTTIME || ' | ' ||  TAUCTIONENDTIME|| ' | ' ||CURBID|| ' | ' ||STATUS);  END LOOP;  CLOSE RESULTSET;  --rollback;  END; |
| Sample Output: |
| --OUTPUT:  --PL/SQL procedure successfully completed.  --  --ID | NAME | CATE | AUCTION STAR TIME |  AUCTION END TIME| CURRENT BID | STATUS  --101 | DATABASES BOOK | BOOK | 03-NOV-16 11.00.00.000000 AM |  12-NOV-16 12.00.00.000000 PM | 1501 | 1 |

|  |  |
| --- | --- |
| NO.9 Functionality: Add Bid To item (Bid On Item) | |
| User wishes to view detailed information about an item they listed above. (For Seller)  OUTPUT VALUE LOOKUP   * 1: Success * -1: Auction has not yet started. * 2: Item has been sold * 3: MaxBidLimit is less than Start Price * 4: User currently is the Winner | |
| SQL Queries: (Excute left one first and then the right one) | |
| set serveroutput on;  declare  SUCC Integer;  begin  ADD\_BID\_PRO(4,106,100,SUCC);  DBMS\_OUTPUT.PUT\_LINE(SUCC);  end; | set serveroutput on;  declare  SUCC Integer;  begin  ADD\_BID\_PRO(4,106,150,SUCC);  DBMS\_OUTPUT.PUT\_LINE(SUCC);  end; |
| SELECT \* FROM harambase\_bid WHERE itemid = 106; | |
| Sample Output: | |
| 1 | 4 |
| -- USERID ITEMID BIDDINGTIME MAXBIDLIMIT  ------------ ---------- ------------------------------- -----------  -- 4 106 15-NOV-16 01.18.37.457000000 AM 100 | |

|  |
| --- |
| NO.10 Functionality: List Items Bid On |
| This page displays every item that the user is bidding on. The item ID, item name, category, the auction start and end time, the start price, the current price, the winner (if there is one) will be displayed on this page. |
| SQL Queries: |
| SELECT Distinct I.ItemID, I.ItemName, I.ItemCategory, I.AuctionStartTime,  I.AuctionEndTime, Harambase\_getPrice\_Func(I.ItemID) AS CurrentBid,  M.Uname AS Winner  FROM Harambase\_BID B, Harambase\_ITEM I, Harambase\_Member M, Harambase\_BID X  WHERE B.UserID = 4 AND B.ItemID = I.ItemID AND M.UserID = GetWinner\_Func(I.ItemID)  ORDER BY I.ITEMID; |
| Sample Output: |
| --ItemID ItemName ItemCategory StartTime  EndTime CurrentBid Winner  -----------------------------------------------------------------------------------  --101 DATABASES BOOK BOOK 03-NOV-16 11.00.00.000000000 AM 12-NOV-16 12.00.00.000000000 PM 151 irahal  --102 CHINSES INSTANT NODDLES FOOD 03-NOV-16 11.00.00.000000000 AM 10-NOV-16 12.00.00.000000000 PM 7 irahal  --104 Abstarct Mathematics BOOK 03-NOV-16 11.00.00.000000000 AM 10-NOV-16 12.00.00.000000000 PM 21 irahal |

|  |
| --- |
| NO.11 Functionality: List Items bought |
| This page displays every item that the user has won. The item ID, item name, category, the auction start and end time, the start price, the sold price, the seller username and the seller email is displayed on this page. The user can also rate the seller from this page. (For Customer) |
| SQL Queries: |
| SELECT Distinct I.ItemID, I.ItemName, I.ItemCategory, I.AuctionStartTime,  I.AuctionEndTime, I.StartPrice, Harambase\_getPrice\_Func(I.ItemID)  AS SoldPrice, M.UName AS SellerUname, M.Email  FROM Harambase\_ITEM I, Harambase\_BID B, Harambase\_MEMBER M  WHERE B.UserID = 4 AND M.UserID = I.SellerID AND B.UserID = GetWinner\_Func(I.ItemID)  ORDER BY I.ITEMID; |
| Sample Output: |
| --ItemID ItemName ItemCategory StartTime EndTime StartPrice SoldPrice SellerUname  Email  --101 DATABASES BOOK BOOK 03-NOV-16 11.00.00.000000000 AM  12-NOV-16 12.00.00.000000000 PM 100 151 slin  [slin@harambase.org](mailto:slin@harambase.org)  --102 CHINSES INSTANT NODDLE FOOD 03-NOV-16 11.00.00.000000000 AM  10-NOV-16 12.00.00.000000000 PM 20 7 mkounn mkounn@harambase.org  --104 Abstarct Mathematics BOOK 03-NOV-16 11.00.00.000000000 AM  10-NOV-16 12.00.00.000000000 PM 20 21 slin slin@harambase.org |

|  |
| --- |
| NO.12 Functionality: Rate Seller or Leave feedback |
| This page is linked to the *List Items Bought* functionality. The Item Id is auto filled while the overall rating, item quality, delivery, and comments can be entered in by the user to leave for the seller to review. This cannot be changed when entered. |
| SQL Queries: |
| SELECT \* FROM HARAMBASE\_FEEDBACK FEED WHERE FEED.ITEMID = 103;  EXEC ADD\_FEEDBACK\_PRO(103, 10, 5, 5, 'VERY GOOD');  SELECT \* FROM HARAMBASE\_FEEDBACK FEED WHERE FEED.ITEMID = 103; |
| Sample Output: |
| --PL/SQL procedure successfully completed.  -- ITEMID OVERALLRATING ITEMQUALITY DELIVERY COMMENTS  -----------------------------------------------------------------  -- 103 10 5 5 VERY GOOD |

|  |
| --- |
| NO.13 Functionality: View my Feedback |
| The view my feedback page displays a table of all the feedback a user has received. The table displays the username of the user who submitted the feedback, as well as the item number, overall rating, item quality, delivery rating and any comments left by the reviewer. |
| SQL Queries: |
| SELECT Distinct M.UName, B.ItemID, F.OverallRating, F.ItemQuality, F.Delivery,  F.Comments  FROM Harambase\_FEEDBACK F, Harambase\_ITEM I, Harambase\_MEMBER M, Harambase\_BID B  WHERE I.SellerID = 3 AND I.ItemID = F.ItemID  AND B.MaxBidLimit = (SELECT Max(MaxBidLimit)  FROM Harambase\_BID B  WHERE I.ItemID = B.ItemID)  AND F.ItemID = B.ItemID AND B.UserID = M.UserID; |
| Sample Output: |
| --Uname ItemID OverallRating ItemQuality Delivery Comments  ---------------------------------------------------------------------  --irahal 104 5 5 4 The NOODLE IS GOOD.  --irahal 101 8 5 4 The BOOK IS GOOD. |

# **PART IV: SQL Code Components**

|  |  |  |
| --- | --- | --- |
| 2 Components: **Sequence** | | |
| **No.** | ***Description*** | **Code** |
| **1** | ***Title:*** USERSEQ  This is the sequence that will automatic generates the next USERID increment by 1.  First it will get the CURMAX from the member table which represents the current maximum USERID. So, the next USERID will be CURMAX + 1. | DECLARE  CURMAX INTEGER;  BEGIN  CURMAX := 1;  SELECT MAX(USERID) INTO CURMAX FROM HARAMBASE\_MEMBER;  IF CURMAX > 1 THEN  EXECUTE IMMEDIATE 'DROP SEQUENCE USERSEQ';  END IF;  CURMAX := CURMAX + 1;  EXECUTE IMMEDIATE 'CREATE SEQUENCE USERSEQ MINVALUE 1 MAXVALUE 1000 INCREMENT BY 1 START WITH '||CURMAX||' NOCACHE ORDER NOCYCLE NOPARTITION' ;  END;  --Do not run the if statement for the first time because there will be no sequence |
| **2** | ***Title:*** ITEMSEQ  This is the sequence that will automatic generates the next ITEMID increment by 1.  First it will get the CURMAX from the member table which represents the current maximum USERID. So, the next USERID will be CURMAX + 1. | DECLARE  CURMAX INTEGER;  BEGIN  CURMAX := 1;  SELECT MAX(ITEMID) INTO CURMAX FROM HARAMBASE\_ITEM;  IF CURMAX > 100 THEN  EXECUTE IMMEDIATE 'DROP SEQUENCE ITEMSEQ';  END IF;  CURMAX := CURMAX + 1;  EXECUTE IMMEDIATE 'CREATE SEQUENCE ITEMSEQ MINVALUE 100 MAXVALUE 1000 INCREMENT BY 1 START WITH '||CURMAX||' NOCACHE ORDER NOCYCLE NOPARTITION' ;  END;  --Do not run the if statement for the first time because there will be no sequence |

|  |  |  |
| --- | --- | --- |
| 2 Components: **Trigger** | | |
| **NO** | **Description** | **Code** |
| **1** | ***Title:*** *GenerateItemID*  This is the trigger that will get the itemid from the ITEMSEQ sequence before it gets insert into the database | Create or Replace Trigger GenerateItemID  BEFORE INSERT ON Harambase\_ITEM  FOR EACH ROW  BEGIN  SELECT ITEMSEQ.NEXTVAL into :new.ItemID FROM dual;  END; |
| **2** | ***Title:*** Generateuserid  This is the trigger that will get the userID from the USERSEQ sequence before it gets insert into the database | CREATE OR REPLACE TRIGGER Generateuserid  BEFORE INSERT ON harambase\_member  FOR EACH ROW  BEGIN  SELECT USERSEQ.nextval INTO :NEW.userid FROM dual;  END; |

|  |  |  |
| --- | --- | --- |
| 5 Components: **Functions** | | |
| **NO** | **Description** | **Code** |
| **1** | ***Title:*** MEMBER\_LOG\_IN\_FUNC  This is the function that will return value of ISLOG. It counts how many maching records in the database (should be 1 or 0) and then put the count result into ISLOG. If ISLOG = 1, it means we find the record and successful login. If ISLOG = 0, it means we have a mismatch somewhere and login failed | create or replace FUNCTION MEMBER\_LOG\_IN\_FUNC  (  USERNAME IN VARCHAR2  , USERPASSWORD IN VARCHAR2  ) RETURN INTEGER AS ISLOG INT := 0;  --CHECK FOR USERNAME AND PASSWORD. T/F = 1/0  BEGIN  SELECT COUNT(\*) INTO ISLOG FROM HARAMBASE\_MEMBER M  WHERE M.UNAME = USERNAME AND M.PASSWORD = USERPASSWORD;  RETURN ISLOG; --RETURN 1 IF COUNT IS 1.  END MEMBER\_LOG\_IN\_FUNC; |
| **2** | ***Title:*** ADMIN\_LOG\_IN\_FUNC  This is the function that will return value of ISLOG. It counts how many maching records in the database (should be 1 or 0) and then put the count result into ISLOG. If ISLOG = 1, it means we find the record and successful login. If ISLOG = 0, it means we have a mismatch somewhere and login failed | create or replace FUNCTION ADMIN\_LOG\_IN\_FUNC  (  USERNAME IN VARCHAR2  , USERPASSWORD IN VARCHAR2  ) RETURN INTEGER AS ISLOG INT := 0;  --CHECK FOR USERNAME AND PASSWORD. T/F = 1/0  BEGIN  SELECT COUNT(\*) INTO ISLOG FROM HARAMBASE\_ADMIN A  WHERE A.UNAME = USERNAME AND A.PASSWORD = USERPASSWORD;  RETURN ISLOG; --RETURN 1 IF COUNT IS 1.    END ADMIN\_LOG\_IN\_FUNC; |
| **3** | ***Title:*** GetPrice\_Func  This function will return the current price of the item. It will return the price of second highest bid + 1 as current bidding price | create or replace Function GetPrice\_Func  (  iID int  ) Return int  AS  price int := 0;  numRows int := 0;  Begin  Select count(\*) Into numRows  From Harambase\_BID B  Where B.ItemID = iID;    If (numRows = 0) Then  RETURN 0;  ElsIf (numRows = 1) Then  Select I.StartPrice + 1 Into price  From Harambase\_Item I  Where I.ItemID = iID;  Else  Select B.MaxBidLimit + 1 Into price  From Harambase\_BID B  Where B.ItemID = iID AND rownum = 1  Order By B.MaxBidLimit;  End If;  Return price;  End Harambase\_getPrice\_Func; |
| **4** | ***Title:*** GETWINNER\_FUNC  This function will return the winner’s id of the input itemid , if not bid on it will return 0 otherwise it will finds the bidder with the highest bid and submitted in the earliest time then return the userid. | create or replace Function GETWINNER\_FUNC  (  item int  ) Return int  AS  winner int := 0;  numMax int := 0;  Begin  Select count(UserID) Into numMax From Harambase\_BID  Where MaxBidLimit = (SELECT Max(X.MaxBidLimit)  FROM Harambase\_BID X  WHERE X.ItemID = item)  AND ItemID = item;  IF (numMax > 1) THEN  Select B.UserID Into winner From Harambase\_BID B  Where B.MaxBidLimit = (SELECT Max(X.MaxBidLimit)  FROM Harambase\_BID X  WHERE X.ItemID = item)  AND ItemID = item  AND B.BIDDINGTIME < (SELECT Y.BIDDINGTIME  FROM HARAMBASE\_BID Y  WHERE Y.MAXBIDLIMIT = MAXBIDLIMIT  AND Y.USERID <> B.USERID  AND Y.ItemID = item);  ElsIf (numMax=0) THEN  winner := 0;  ELSE  Select UserID Into winner From Harambase\_BID  Where MaxBidLimit = (SELECT Max(X.MaxBidLimit)  FROM Harambase\_BID X  WHERE X.ItemID = item)  AND ItemID = item;  End If;  Return winner;    END GETWINNER\_FUNC; |
| **5** | ***Title:*** GET\_STATUS\_FUNC  This function will return the item’s STATUS.  Possible output of variable STATUS:   1. **1:** the item has been sold 2. **0:** the item is still on Auction 3. **-1**: the Auction has not yet started. | create or replace function get\_status\_func  (  iid int  ) return int AS  STATUS int := 0;  endTime date := TO\_TIMESTAMP('01/JAN/1900 12:00:00','DD/MM/YYYY HH:MI:SS');  startTime date := TO\_TIMESTAMP('01/JAN/1900 11:00:00','DD/MM/YYYY HH:MI:SS');  begin  Select I.AuctionEndTime Into endTime From Harambase\_Item I Where I.ItemID = iid;  Select I.AuctionStartTime Into startTime From Harambase\_Item I Where I.ItemID = iid;    If (endTime < CURRENT\_TIMESTAMP) Then  STATUS := 1; --SOLD  ELSE  STATUS := 0; --ON AUCTION  End If;    IF (startTime > CURRENT\_TIMESTAMP) THEN  STATUS := -1; --NOT ON AUCTION  END IF;    RETURN STATUS;    end get\_status\_func; |

|  |  |  |
| --- | --- | --- |
| 6 *Components:* Procedures | | |
| NO | **Description** | **Code** |
| 1 | ***Title:***  ADD\_BID\_PRO  This procedure will add a new bidding object into the database.  The variable SUCC is out variable which has following possible results:   1. 1: success 2. 2: item has been sold 3. -1: Auction has not started 4. 3: The MAXBIDLIMIT is smaller than the start price 5. 4: the user is currently the winner. | CREATE or replace PROCEDURE ADD\_BID\_PRO  (  USERID IN NUMBER  , TID IN NUMBER  , MAXBIDLIMIT IN NUMBER  , SUCC OUT INTEGER  ) AS  STATUS INT := 0;  STARTPRICE NUMBER := 0;  CURRBID NUMBER := 0;  WINNERID NUMBER := 0;  CT INT := 0;  BEGIN  SELECT get\_status\_func(TID) INTO STATUS FROM DUAL;  SELECT HARAMBASE\_GETPRICE\_FUNC(TID) INTO CURRBID FROM DUAL;  SELECT ITEM.STARTPRICE INTO STARTPRICE FROM HARAMBASE\_ITEM ITEM  WHERE ITEM.ITEMID = TID;  SELECT GETWINNER\_FUNC(TID) INTO WINNERID FROM DUAL;    IF STATUS = 0 AND STARTPRICE < MAXBIDLIMIT AND USERID <> WINNERID THEN  SELECT COUNT(\*) INTO CT FROM HARAMBASE\_BID BID WHERE BID.USERID = USERID;  IF CT = 1 THEN  DELETE FROM team2.HARAMBASE\_BID BID  WHERE BID.USERID = USERID;  END IF;  INSERT INTO team2.HARAMBASE\_BID  VALUES (USERID, TID, CURRENT\_TIMESTAMP, MAXBIDLIMIT);  SUCC := 1; --SUCCESS  ELSIF STATUS = 1 THEN SUCC := 2;  --ITEM HAS BEEN SOLD  ELSIF STATUS = -1 THEN SUCC := -1;  --THE AUCTION HAS NOT STARTED YET  ELSIF STATUS = 0 AND STARTPRICE >= MAXBIDLIMIT THEN SUCC := 3;  --MAXLIMIT REJECTED  ELSIF STATUS = 0 AND STARTPRICE < MAXBIDLIMIT  AND USERID = WINNERID  THEN SUCC := 4; --WINNER REJECTED  END IF;    END ADD\_BID\_PRO; |
| 2 | ***Title:***  ADD\_ITEM\_PRO  This procedure will add a new item into the database after the trigger generates the new ITEMID. | create or replace PROCEDURE ADD\_ITEM\_PRO  (  ITEMID IN NUMBER  , ITEMNAME IN VARCHAR2  , ITEMCATEGORY IN VARCHAR2  , STARTPRICE IN NUMBER  , DESCRIPTION IN VARCHAR2  , SELLERID IN NUMBER  , AUCTIONSTARTTIME IN TIMESTAMP  , AUCTIONENDTIME IN TIMESTAMP  ) AS  BEGIN  INSERT INTO team2.HARAMBASE\_ITEM VALUES  (ITEMID,ITEMNAME,ITEMCATEGORY,STARTPRICE,  DESCRIPTION,SELLERID,AUCTIONSTARTTIME,AUCTIONENDTIME);  END ADD\_ITEM\_PRO; |
| 3 | ***Title:***  ADD\_FEEDBACK\_PRO  This procedure will add a new feedback object into the database by the specified variable values. | create or replace PROCEDURE ADD\_FEEDBACK\_PRO  (  ITEMID IN NUMBER  , OVERALLRATING IN NUMBER  , ITEMQUALITY IN NUMBER  , DELIVERY IN NUMBER  , COMMENTS IN VARCHAR2  ) AS  BEGIN  INSERT INTO team2.HARAMBASE\_FEEDBACK VALUES (ITEMID,OVERALLRATING,ITEMQUALITY,DELIVERY,COMMENTS);  END ADD\_FEEDBACK\_PRO; |
| 4 | ***Title:***  ADD\_USER\_PRO  This procedure will add a new user into the database after the trigger generates the new USERID .  It will have an output integer STATUS which has following possible values:   1. 0: success 2. 1: Violating rules possibly UNAME is duplicated. | create or replace PROCEDURE ADD\_USER\_PRO  (  USERID IN NUMBER  , NAME IN VARCHAR2  , EMAIL IN VARCHAR2  , FNAME IN VARCHAR2  , LNAME IN VARCHAR2  , PASSWORD IN VARCHAR2  , CREATORID IN NUMBER  , ISBUYER IN NUMBER  , ISELLER IN NUMBER  , STATUS OUT INTEGER  ) AS  BEGIN  STATUS := 0;  SELECT COUNT(\*) INTO STATUS FROM HARAMBASE\_MEMBER M WHERE M.UNAME = NAME;  IF STATUS = 0 THEN  INSERT INTO team2.HARAMBASE\_MEMBER VALUES (USERID,NAME,EMAIL,FNAME,LNAME,PASSWORD,CREATORID,ISBUYER,ISELLER);  END IF;  END ADD\_USER\_PRO; |
| 5 | ***Title:***  SEARCH\_ITEM\_PRO  Searches for an item based on the given parameters and returns a RESULTSET that contains the results as a form of a table.  The search PROCEDURE contains following functionality:  1. Search by item id alone,  2. Search by keyword alone,  3. Search by keyword and category,  4. Search by keyword and current bid range,  5. Search by keyword and auction time period,  6. Search by keyword, category and current bid range,  7. Search by keyword, category and auction time period,  8. Search by keyword, current bid range and auction time period,  9. Search by keyword, category, current bid range and auction time period.  10. Inexact search on item name. | create or replace PROCEDURE SEARCH\_ITEM\_PRO  (  TID IN NUMBER  , KEYWORD IN VARCHAR2  , TCATEGORY IN VARCHAR2  , TAUCTIONSTARTTIME IN TIMESTAMP  , TAUCTIONENDTIME IN TIMESTAMP  , CURBIDMIN IN NUMBER  , CURBIDMAX IN NUMBER  , RESULTSET OUT SYS\_REFCURSOR  ) AS  BEGIN    IF TID IS NOT NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE HI.ITEMID = TID;  ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NOT NULL  AND CURBIDMIN IS NOT NULL  AND CURBIDMAX IS NOT NULL  AND TAUCTIONSTARTTIME IS NOT NULL  AND TAUCTIONENDTIME IS NOT NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE (UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME))  AND TCATEGORY = HI.ITEMCATEGORY  AND HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) <= CURBIDMAX  AND CURBIDMIN <= HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID)  AND TAUCTIONSTARTTIME <= HI.AUCTIONSTARTTIME  AND TAUCTIONENDTIME >= HI.AUCTIONENDTIME;  ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NULL  AND CURBIDMIN IS NOT NULL  AND CURBIDMAX IS NOT NULL  AND TAUCTIONSTARTTIME IS NOT NULL  AND TAUCTIONENDTIME IS NOT NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE (UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME))  AND HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) <= CURBIDMAX  AND CURBIDMIN <= HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID)  AND TAUCTIONSTARTTIME <= HI.AUCTIONSTARTTIME  AND TAUCTIONENDTIME >= HI.AUCTIONENDTIME;    ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NOT NULL  AND CURBIDMIN IS NULL  AND CURBIDMAX IS NULL  AND TAUCTIONSTARTTIME IS NOT NULL  AND TAUCTIONENDTIME IS NOT NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE (UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME))  AND TCATEGORY = HI.ITEMCATEGORY  AND TAUCTIONSTARTTIME <= HI.AUCTIONSTARTTIME  AND TAUCTIONENDTIME >= HI.AUCTIONENDTIME;    ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NOT NULL  AND CURBIDMIN IS NOT NULL  AND CURBIDMAX IS NOT NULL  AND TAUCTIONSTARTTIME IS NULL  AND TAUCTIONENDTIME IS NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE (UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME))  AND TCATEGORY = HI.ITEMCATEGORY  AND HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) <= CURBIDMAX  AND CURBIDMIN <= HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID);    ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NULL  AND CURBIDMIN IS NOT NULL  AND CURBIDMAX IS NOT NULL  AND TAUCTIONSTARTTIME IS NULL  AND TAUCTIONENDTIME IS NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE (UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME))  AND HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) <= CURBIDMAX  AND CURBIDMIN <= HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID);  ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NULL  AND CURBIDMIN IS NULL  AND CURBIDMAX IS NULL  AND TAUCTIONSTARTTIME IS NOT NULL  AND TAUCTIONENDTIME IS NOT NULL THEN    OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE (UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME))  AND TAUCTIONSTARTTIME <= HI.AUCTIONSTARTTIME  AND TAUCTIONENDTIME >= HI.AUCTIONENDTIME;    ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NOT NULL  AND CURBIDMIN IS NULL  AND CURBIDMAX IS NULL  AND TAUCTIONSTARTTIME IS NULL  AND TAUCTIONENDTIME IS NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE (UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME))  AND TCATEGORY = HI.ITEMCATEGORY;    ELSIF KEYWORD IS NOT NULL  AND TCATEGORY IS NULL  AND CURBIDMIN IS NULL  AND CURBIDMAX IS NULL  AND TAUCTIONSTARTTIME IS NULL  AND TAUCTIONENDTIME IS NULL THEN  OPEN RESULTSET FOR  SELECT HI.ITEMID, HI.ITEMNAME,  HI.ITEMCATEGORY, HI.AUCTIONSTARTTIME, HI.AUCTIONENDTIME,  HARAMBASE\_GETPRICE\_FUNC(HI.ITEMID) AS CURRENTBID,  GET\_STATUS\_FUNC(HI.ITEMID) AS STATUS  FROM HARAMBASE\_ITEM HI  WHERE UPPER(HI.ITEMNAME) LIKE '%'||UPPER(KEYWORD)||'%'  OR SOUNDEX(KEYWORD) = SOUNDEX(HI.ITEMNAME);  END IF;    END SEARCH\_ITEM\_PRO; |
| 6 | ***Title:*** UPDATE\_PROFILE\_PRO  This procedure will update the user’s profile. It will have an output integer which has following possible values:   1. 1: success 2. 0: Violating rules possible 3. -1: New passwords do not match 4. -2: Wrong old password | create or replace PROCEDURE UPDATE\_PROFILE\_PRO  (  URID IN NUMBER  , USERNAME IN VARCHAR2  , USEREMAIL IN VARCHAR2  , USERFNAME IN VARCHAR2  , USERLNAME IN VARCHAR2  , USEROLDPASSWORD IN VARCHAR2  , USERNEWPASSWPRD\_1 IN VARCHAR2  , USERNEWPASSWORD\_2 IN VARCHAR2  , STATES OUT INTEGER  ) AS  BEGIN  STATES := 0;  IF USERNAME IS NOT NULL THEN  UPDATE team2.HARAMBASE\_MEMBER set UNAME = USERNAME WHERE USERID = URID;  STATES := 1;  END IF;    IF USEREMAIL IS NOT NULL THEN  UPDATE team2.HARAMBASE\_MEMBER set EMAIL = USEREMAIL WHERE USERID = URID;  STATES := 1;  END IF;    IF USERFNAME IS NOT NULL THEN  UPDATE team2.HARAMBASE\_MEMBER set FNAME = USERFNAME WHERE USERID = URID;  STATES := 1;  END IF;    IF USERLNAME IS NOT NULL THEN  UPDATE team2.HARAMBASE\_MEMBER set LNAME = USERLNAME WHERE USERID = URID;  STATES := 1;  END IF;    IF USEROLDPASSWORD IS NOT NULL AND USERNEWPASSWPRD\_1 IS NOT NULL AND USERNEWPASSWORD\_2 IS NOT NULL THEN  IF USERNEWPASSWPRD\_1 = USERNEWPASSWORD\_2 THEN  SELECT COUNT(\*) INTO STATES FROM HARAMBASE\_MEMBER M WHERE M.USERID = URID AND M.PASSWORD = USEROLDPASSWORD;  IF STATES > 0 THEN  UPDATE team2.HARAMBASE\_MEMBER set PASSWORD = USERNEWPASSWPRD\_1 WHERE USERID = URID;  STATES := 1;  END IF;  STATES := -2;  ELSE  STATES := -1;  END IF;  END IF;  -- 1:sucess  -- 0:Violating rules possible  -- -1:New password does not match  -- -2:Wrong old password  END UPDATE\_PROFILE\_PRO; |

|  |  |  |
| --- | --- | --- |
| 2 *Components:* views | | |
| NO | **Description** | **Code** |
| 1 | ***Title:*** SALES\_SUMMARY\_REPORT  **Creates or replaces a view titled** *SALES\_SUMMARY\_REPORT* **which selects all the items sold and sorts them by category and then by item id. A commission field is also calculated from 5% of the final selling price** | CREATE OR REPLACE FORCE VIEW SALES\_SUMMARY\_REPORT AS  SELECT ITEM.ITEMCATEGORY, ITEM.ITEMID, ITEM.ITEMNAME, HARAMBASE\_GETPRICE\_FUNC(ITEM.ITEMID) AS FINAL\_SELLING\_PRICE,  HARAMBASE\_GETPRICE\_FUNC(ITEM.ITEMID)\*0.05 AS COMMISSION  FROM HARAMBASE\_ITEM ITEM  WHERE GET\_STATUS\_FUNC(ITEM.ITEMID) = 1  ORDER BY ITEM.ITEMCATEGORY; |
| 2 | ***Title:*** Overall\_Commission\_View  **Creates or replaces a view titled** *OVERALL\_COMMISSION\_VIEW* **which selects all the user ids and displays their basic information such as user id, user name, first name, last name, and email. It also displays a SELLER RATING and COMMISSION field which are calculated by averaging out the users OVERALL RATING from their feedback and the sum of the commissions from their sold items.** | CREATE OR REPLACE VIEW Overall\_Commission\_View AS  SELECT HM.UserID as userID, HM.Uname as User\_Name,  HM.Fname as First\_Name,  HM.Lname as Last\_Name, HM.Email as Email,  AVG(HF.OverallRating) AS Seller\_Rating,  SUM(SR.COMMISSION) AS COMMISSIONS  FROM Harambase\_Feedback HF, Harambase\_MEMBER HM,  SALES\_SUMMARY\_REPORT SR  WHERE HM.UserID IN (Select Item.SellerID  From Harambase\_Item Item  WHERE ITEM.ITEMID = SR.ITEMID AND ITEM.ITEMID = HF.ITEMID)    Group By HM.UserID, HM.Uname, HM.Fname, HM.Lname, HM.Email  ORDER BY HM.USERID; |

# **PART V: Highlighted Problems**

* Some of the problems that our team faced was that we thought this phase would be a lot easier than it was!
* the SQL Developer is always a bit unresponsive and bugging at times. These issues needed great patience to overcome.
* One problem our team faced during this phase was that SQL Developer was not being responsive at times.

# **PART VI: Task Decomposition**

|  |  |
| --- | --- |
| **Name** | **Task Assigned and Completed** |
| Shilei Lin | For this phase I wrote the Queries/Functions/Procedures for the follow functionalities: Admin Log-In, View Users, Add User, View Sales Summary Report, Overall Commission Report, Member Log-In, Update Profile, Add Bid On, Add Item, and Search For Items. I also helped my teammates debug their functionalities. |
| Matthew Kounniyom | For this phase I helped write all the Create and Insert statements to create our database. Also, I wrote the Query for the Show Item Info functionality and put this report together. I personally did not do much in this phase due to not fully understanding what was going on because I was sick. My partners helped guide me to understanding PL/SQL. |
| Joshua Folkerds | For this phase I helped write all the Create and Insert statements to create our database. I wrote the SQL queries and the verbose descriptions for the bid on item, show items bid on, list items bought, rate seller, and view my feedback functionalities. I also wrote the GetWinner\_Func() and the Harambase\_GetPrice\_Func() functions. I wrote the GenerateItemID and the GenerateUserID triggers. Throughout this phase, I also helped my teammates to debug their own functionalities. |

# **PART VII: Meeting Minutes**

|  |  |
| --- | --- |
| DatE | Details |
| 10/16 | We met to discuss what we plan on doing for this phase, and worked on updating our diagrams from phase II. |
| 10/20 | \*Shilei believes he met with Imad about the diagrams\* |
| 10/23 | We started writing the create and insert statements for the database. |
| 10/27 | Finished writing the create statements and finalized our inserts with MEANINGFUL data! |
| 10/30 | Started our SQL programming with the little knowledge that we had. Mostly discussed the next few days. |
| 11/03 | We dived farther into SQL programming and divided the tasks evenly to have something done for the next meeting. Meeting ended early due to Matt not feeling well. |
| 11/06 | \*Matt: No Call, No Show\* Because I (Matt) am writing the report, I have no idea what they did this day. I believe they just worked on SQL programming. |
| 11/10 | We got together and worked on our functionalities and discussed logic and helped each other out. |
| 11/13 | Matt started writing the report, and Shilei and Josh worked more on some complicated queries. We also talked about the requirements of this phase again. |
| 11/14 | We met and checked over our queries one final time, and looking over the report. |