Introduction

Online Auction System Description: This system is designed to implement the basic functionalities of an Online Auction system where buyers and sellers participate in the sale of advertised items.

Users are divided into three different account types: buyer, seller, and administrator. Members may create accounts, and can be either a buyer or a seller. This label will determine the account type when the account is created. Upon login, the user will be prompted to enter their username and login. In the case of an error in either of the fields, the system will prompt the user with an error message. The login page will also provide users the option to create a new account, which will also prompt the user to specify their account type as a requirement to create their account. Upon account creation, buyers will be prompted to provide the following information: email address, name, password, billing address, shipping address, payment information, and phone number. Sellers will be prompted to provide email address, name, password, billing address, bank account number, routing number, and phone number. Once logged in, the users will have the option to log out of the system.

Buyers may search through all the available products for auction, and may place their bids on their desired product. The buyers will be able to view the starting bid for the given product, as well as the current highest bid. Additionally, the buyers will be permitted to update their bids on products to beat other bids. When each bid is made, the system stores the bid time and the bid price. When the time allotted to the auction ends, the system declares the winner of the auction and proceeds with the transaction between the buyer and the seller. These buyers may also view all their current active bids as well as their expired bids.

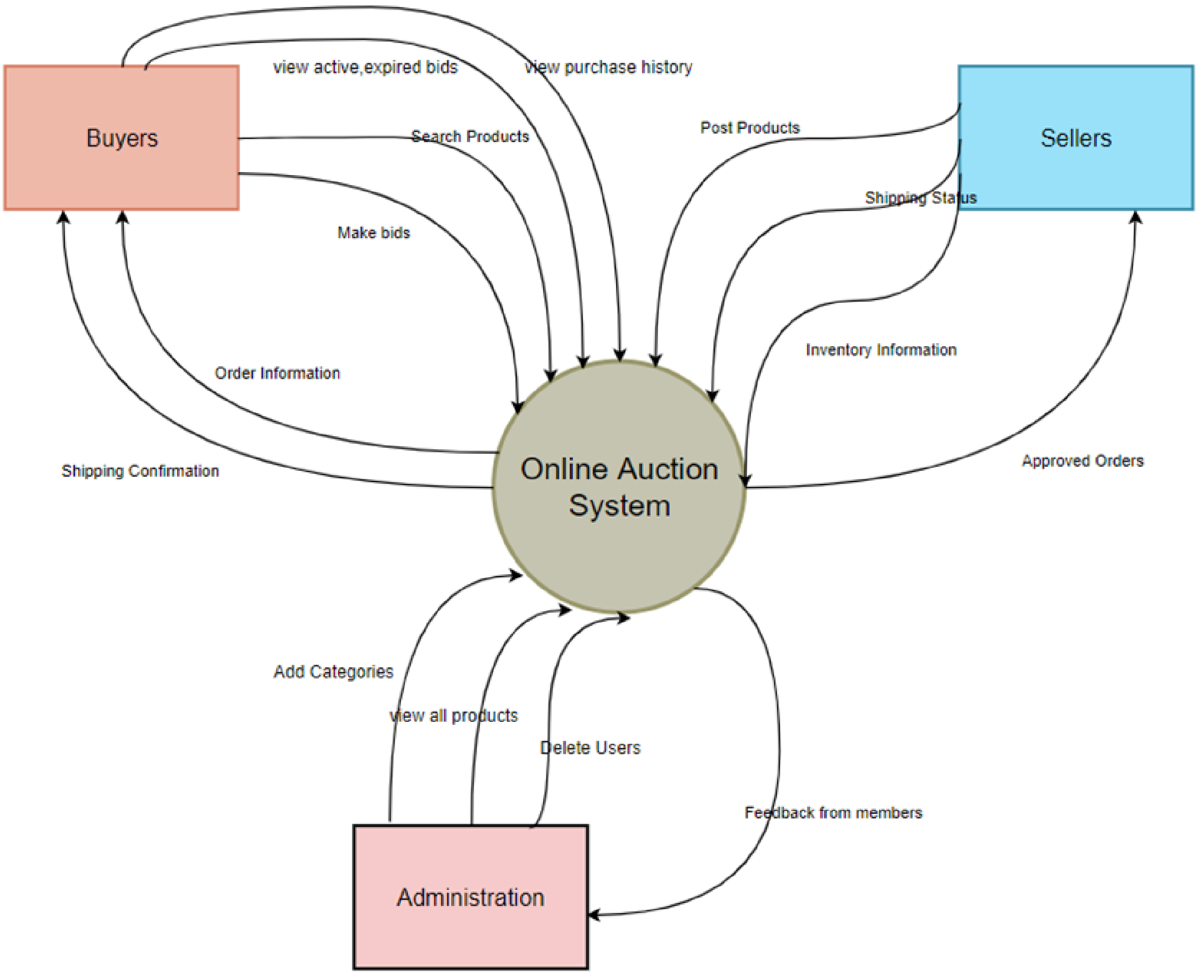
Sellers will similarly be allowed to search through all items available for auction and display the result. Additionally, sellers may post ads for the products they desire to sell, and will be able to see both their active and expired bids. Once the auction is over, the system sends the seller the customer order and the seller begins processing/shipping the auctioned product.

When the item is placed for sale, a unique item number is assigned to it by the system. All items for sale are categorized based on a fixed classification hierarchy which are defined by the administrators.

Administrators will be able to manage (view, control, search, delete) all product ads posted on the website as well as contact messages. Admins are also allowed to view feedback from buyers and sellers. Admins also serve as regulators to view and delete users in case of violations.

System Requirements

Context Diagram (System Architecture):



Interface Requirements

I1. User(Buyer/Seller) login/Admin Login capabilities

I2. Consistency of layout/design across the website. (e.g. having a header)

I3. Clarity of layout and design: Should be easy to navigate and find functions.

I4. Responsiveness of website: Display works across multiple devices (e.g. works on different screen sizes and display interactions with the user.)

I5. Familiarity: The page should do what the user expects. (e.g. a magnifying glass would logically represent a search option)

I6. Aesthetics (e.g. page colors, images, page elements, sections are neatly aligned)

I7. Error Analysis. Validation/returning error messages to the user.

I8. Ease of Access to ‘Help’ (Documentation)

Functional Requirements

Login Functional Requirements:

FR1. System will allow the user to login.
FR2. System will verify the username and password.
FR3. System will not allow the user to login with invalid username or password.
FR4. System will be able to remember username and password.
FR5. System will allow users to create an account.
FR6. System will enable users to logout of their accounts
FR7. System should allow sellers to have accounts where they will receive the customer orders and respond to them.

Browsing Functional Requirements

FR8. System will allow the user to search products that are available for auction and shall display the result.
FR9. System will allow the bidder to bid on the desired product.
FR10. System will allow users to post the ad for the product they want to sell.
FR11. System will allow users to view their active bids (that are in progress).
FR12. System will allow the user to view their expired bids.
FR13. System will allow the user to view their purchase history.

Administrator Functional requirements:

F14. Admin can view all the products.
F15. Admin can control, delete and search any product.
F16. Admin can view the feedback sent from users.
F17. Admin can add categories in the database.
F18. Admin can view all the users
F19. Admin has the right to delete any users in case of violations of the website policies.
F20. Admin can view the contact messages.

Additional Functional Requirements:

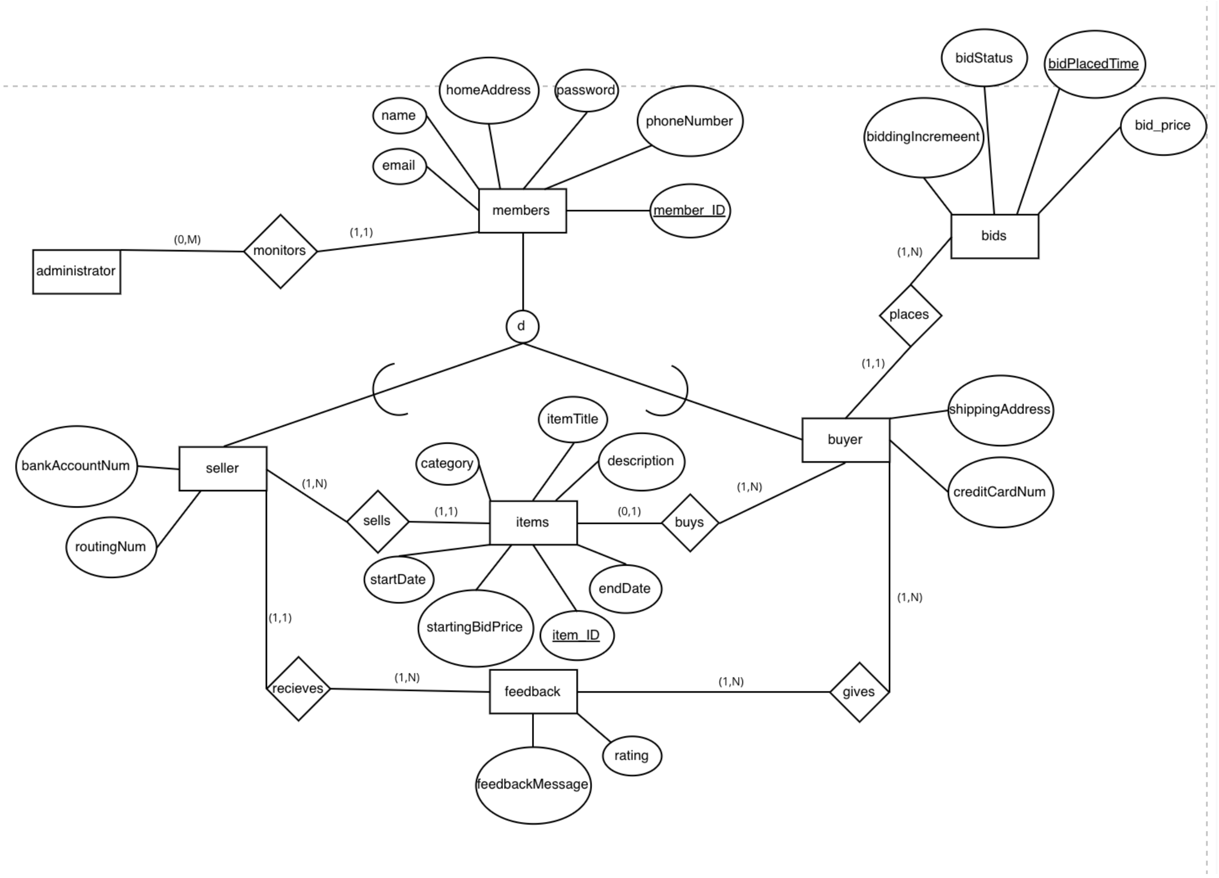
F21. Buyer and Seller can record feedback regarding transactions
F22. Require specification of account type (Buyer/Seller/Administrator) at creation.
F23. System will allow users to enter payment information.
F24. Sellers when putting an item up for auction may set a time limit for the auction duration.

Non-Functional Requirements

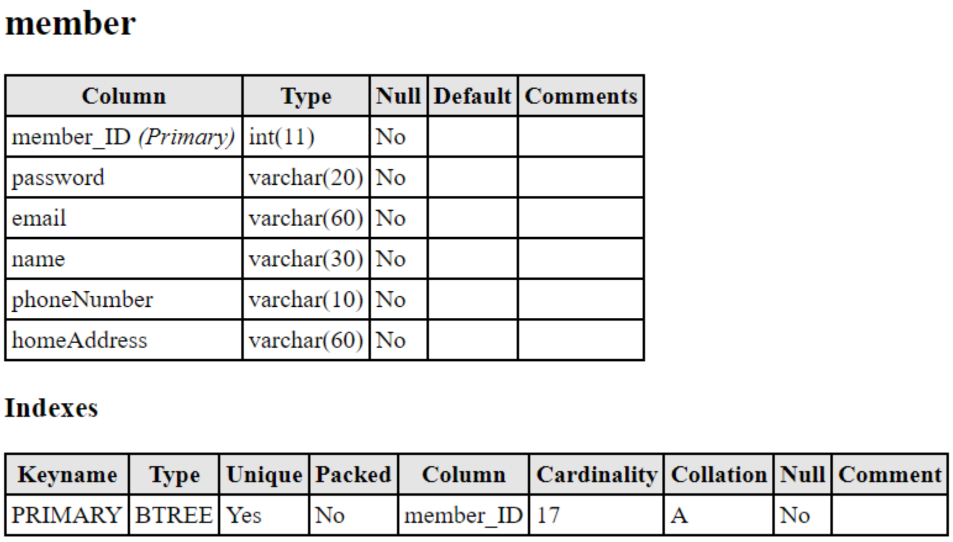
NF1. Each search query should take a reasonable amount of time (example: takes no longer than 10 seconds).
NF2. Database should be compatible with various hardware
NF3. Product should be easy to use for the users with different levels of computer skills
NF4. Database should be scalable
NF5. Product should be secure
NF6. Database should be capable to handle multiple users

Conceptual Design of the Database

ER Diagram:



Data Dictionary and Constraints:



member

Column	Type	Null	Default	Comments
member_ID *(Primary)*	int(11)	No		
password	varchar(20)	No		
email	varchar(60)	No		
name	varchar(30)	No		
phoneNumber	varchar(10)	No		
homeAddress	varchar(60)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	member_ID	17	A	No	

administrator

Column	Type	Null	Default	Comments
member_ID *(Primary)*	int(11)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	member_ID	3	A	No	
FK_admin	BTREE	No	No	member_ID	3	A	No	

buyer

Column	Type	Null	Default	Comments
member_ID *(Primary)*	int(11)	No		
shippingAddress	varchar(60)	No		
creditCardNum	varchar(16)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	member_ID	8	A	No	
FK_Buyer	BTREE	No	No	member_ID	8	A	No	

seller

Column	Type	Null	Default	Comments
member_ID *(Primary)*	int(11)	No		
bankAccountNum	varchar(12)	No		
routingNum	varchar(9)	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	member_ID	4	A	No	
FK_seller	BTREE	No	No	member_ID	4	A	No	

items

Column	Type	Null	Default	Comments
item_ID *(Primary)*	int(11)	No		
seller_ID	int(11)	No		
startingBidPrice	int(11)	No		
description	text	No		
startDate	timestamp	No	CURRENT_TIMESTAMP	
endDate	timestamp	No	CURRENT_TIMESTAMP	
category	varchar(30)	No		
itemTitle	varchar(30)	No		

Indexes

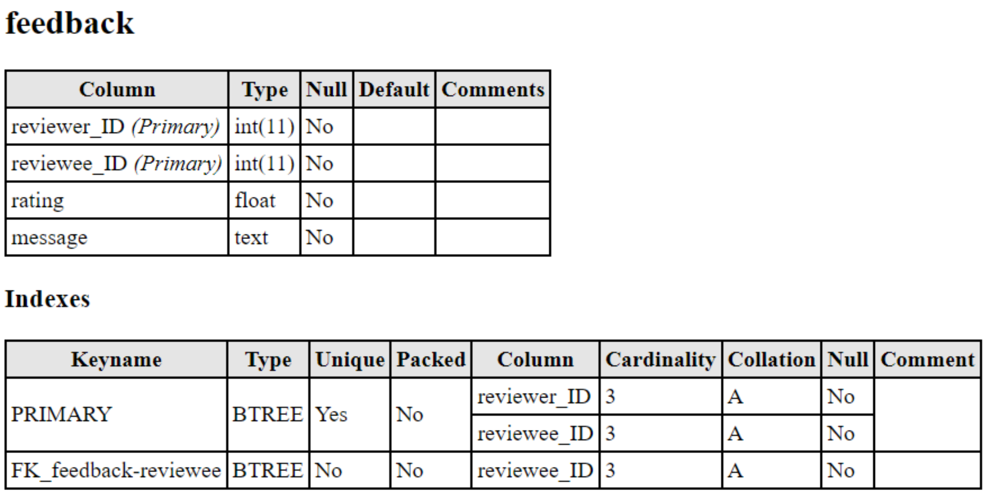
Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	item_ID	4	A	No	
FK_item	BTREE	No	No	seller_ID	4	A	No	

bids

Column	Type	Null	Default	Comments
buyer_ID *(Primary)*	int(11)	No		
seller_ID *(Primary)*	int(11)	No		
item_ID *(Primary)*	int(11)	No		
bidPlacedTime *(Primary)*	timestamp	No	CURRENT_TIMESTAMP	
bidStatus	tinyint(1)	No		
bidPrice	float	No		
bidIncrement	float	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	buyer_ID	4	A	No	
				seller_ID	8	A	No	
				item_ID	8	A	No	
				bidPlacedTime	8	A	No	
FK_bids-seller	BTREE	No	No	seller_ID	4	A	No	
FK_bids-item	BTREE	No	No	item_ID	4	A	No	



feedback

Column	Type	Null	Default	Comments
reviewer_ID *(Primary)*	int(11)	No		
reviewee_ID *(Primary)*	int(11)	No		
rating	float	No		
message	text	No		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	reviewer_ID	3	A	No	
				reviewee_ID	3	A	No	
FK_feedback-reviewee	BTREE	No	No	reviewee_ID	3	A	No	

Logical Database Schema

Relations:

Member [MemberID, Name, Email, Password, phoneNumber, homeAddress]

Buyer [BuyerID (FK), shippingAddress, creditCardNum,]

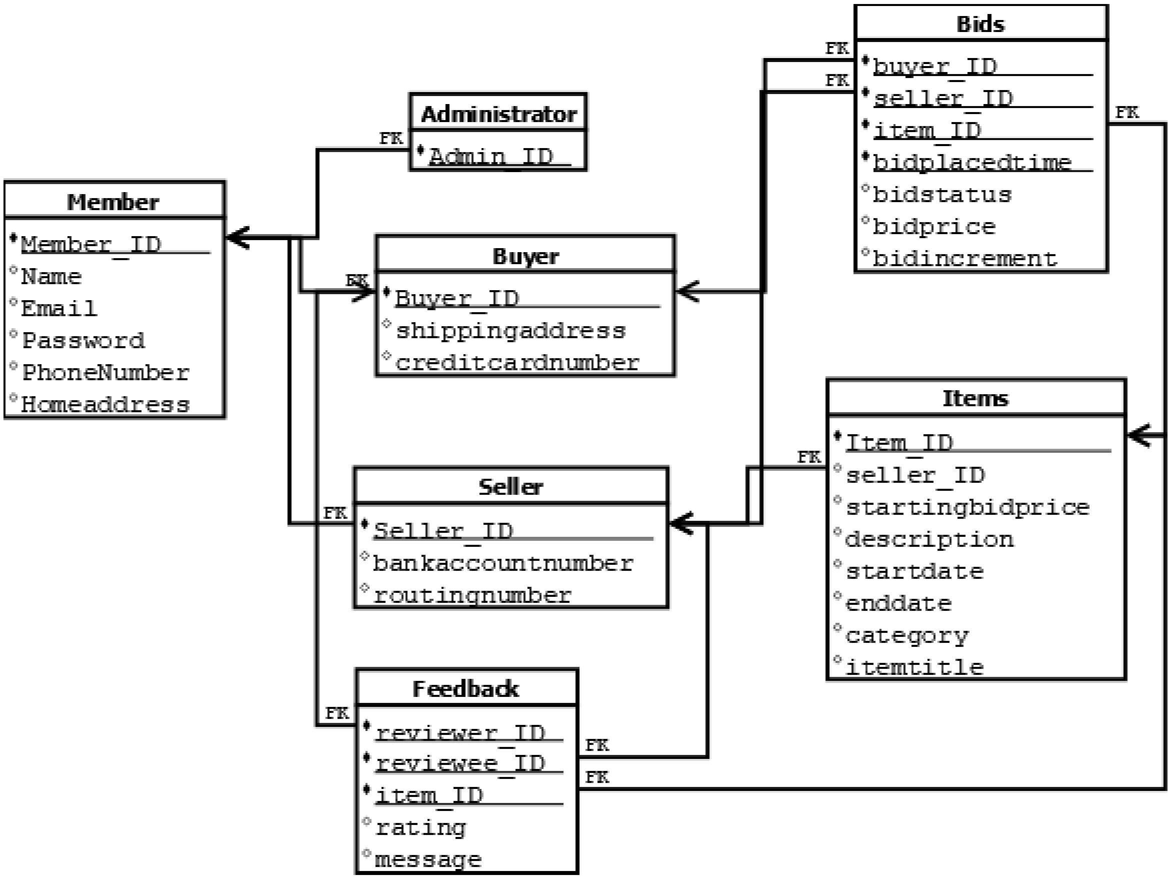
Seller [SellerID (FK), bankAccountNum, routingNum]

Administrator [AdminID (FK)]

Items [ItemID, description, SellerID (FK), itemTitle, category, startingBidPrice, startDate, endDate]

Bids [BuyerID (FK), SellerID (FK), ItemID (FK), bidStatus, bidPlacedTime, bidPrice, biddingIncrement]

Feedback [reviewerID (FK), revieweeID (FK), items (FK), rating, feedbackMessage]



SQL Statements:

Member Table:

```
CREATE TABLE `member` (
  `member_ID` int(11) NOT NULL,
  `password` varchar(20) NOT NULL,
  `email` varchar(60) NOT NULL,
  `name` varchar(30) NOT NULL,
  `phoneNumber` varchar(10) NOT NULL,
  `homeAddress` varchar(60) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `member`
  ADD PRIMARY KEY (`member_ID`);
```

Administrator Table:

```
CREATE TABLE `administrator` (
  `member_ID` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `administrator`
  ADD PRIMARY KEY (`member_ID`),
  ADD KEY `FK_admin` (`member_ID`);
```

```
ALTER TABLE `administrator`
  ADD CONSTRAINT `FK_admin` FOREIGN KEY (`member_ID`) REFERENCES `member` (`member_ID`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Seller Table:

```
CREATE TABLE `seller` (
  `member_ID` int(11) NOT NULL,
  `bankAccountNum` varchar(12) NOT NULL,
  `routingNum` varchar(9) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `seller`
  ADD PRIMARY KEY (`member_ID`),
  ADD KEY `FK_seller` (`member_ID`);


ALTER TABLE `seller`
  ADD CONSTRAINT `FK_seller` FOREIGN KEY (`member_ID`) REFERENCES `member` (`member_ID`);
```

Buyer table:

```
CREATE TABLE `buyer` (
  `member_ID` int(11) NOT NULL,
  `shippingAddress` varchar(60) NOT NULL,
  `creditCardNum` varchar(16) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
ALTER TABLE `buyer`
  ADD PRIMARY KEY (`member_ID`),
  ADD KEY `FK_Buyer` (`member_ID`);


ALTER TABLE `buyer`
  ADD CONSTRAINT `FK_Buyer` FOREIGN KEY (`member_ID`) REFERENCES `member` (`member_ID`);
```

Items Table:

```
CREATE TABLE `items` (
  `item_ID` int(11) NOT NULL,
  `seller_ID` int(11) NOT NULL,
  `startingBidPrice` int(11) NOT NULL,
  `description` text NOT NULL,
  `startDate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `endDate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `category` varchar(30) NOT NULL,
  `itemTitle` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `items`
  ADD PRIMARY KEY (`item_ID`),
  ADD KEY `FK_item` (`seller_ID`);
```

```
ALTER TABLE `items`
  ADD CONSTRAINT `FK_item` FOREIGN KEY (`seller_ID`) REFERENCES `member` (`member_ID`);
```

Bids Table:

```
CREATE TABLE `bids` (
  `buyer_ID` int(11) NOT NULL,
  `seller_ID` int(11) NOT NULL,
  `item_ID` int(11) NOT NULL,
  `bidPlacedTime` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  `bidStatus` tinyint(1) NOT NULL,
  `bidPrice` float NOT NULL,
  `bidIncrement` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `bids`
  ADD PRIMARY KEY (`buyer_ID`,`seller_ID`,`item_ID`,`bidPlacedTime`),
  ADD KEY `FK_bids-seller` (`seller_ID`),
  ADD KEY `FK_bids-item` (`item_ID`);


ALTER TABLE `bids`
  ADD CONSTRAINT `FK_bids-buyer` FOREIGN KEY (`buyer_ID`) REFERENCES `buyer` (`member_ID`),
  ADD CONSTRAINT `FK_bids-item` FOREIGN KEY (`item_ID`) REFERENCES `items` (`item_ID`),
  ADD CONSTRAINT `FK_bids-seller` FOREIGN KEY (`seller_ID`) REFERENCES `seller` (`member_ID`);
```

Feedback Table:

```
CREATE TABLE `feedback` (

  `reviewer_ID` int(11) NOT NULL,

  `reviewee_ID` int(11) NOT NULL,

  `Item` int(11) NOT NULL,

  `rating` float NOT NULL,

  `message` text NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;


ALTER TABLE `feedback`

  ADD PRIMARY KEY (`reviewer_ID`,`reviewee_ID`),

  ADD KEY `FK_feedback-reviewee` (`reviewee_ID`),

  ADD KEY `Item` (`Item`);


ALTER TABLE `feedback`

  ADD CONSTRAINT `FK_feedback-Item` FOREIGN KEY (`Item`) REFERENCES `items` (`item_ID`),

  ADD CONSTRAINT `FK_feedback-reviewee` FOREIGN KEY (`reviewee_ID`) REFERENCES `seller` (`member_ID`),

  ADD CONSTRAINT `FK_feedback-reviewer` FOREIGN KEY (`reviewer_ID`) REFERENCES `buyer` (`member_ID`);
```

Expected Database Operations:

INSERT (CREATE):

- members into the ‘member’ table
- members into the ‘seller’ table
- members into the ‘buyer’ table
- Items into the ‘items’ table
- Bids into the ‘bids’ table
- Messages into the ‘feedback’ table
- Members to the ‘admin’ table, but not supported from the interface for security

UPDATE:

- Member info in the ‘member’ table
- info in the ‘seller’ table
- info in the ‘buyer’ table
- Item info in the ‘items’ table

DELETE:

- Members from the ‘member’ table (done by admins)
- Info in the ‘seller’ table
- Info in the ‘buyer’ table
- Items from the ‘items’ table
- Removing admins from the ‘admin’ table, but not supported from the interface for security

SELECT (READ):

- Listing members from the ‘members’ table
- Listing members from the ‘seller’ table
- Listing members from the ‘buyer’ table
- Listing items from the ‘items’ table
- Viewing messages from the ‘feedback’ table

Estimated Data Volumes:

The number of members are not restricted, nor are the amount of items a seller/buyer can sell/buy. The amount of feedback messages are unrestricted, but estimated to be one per completed transaction. The number of bids placed on an item is expected to be high, moreso when the time frame for the transaction specified by the seller is large. There should be significantly fewer admins than buyers/sellers.

Functional Dependencies and Database Normalization

Set of Functional Dependencies:

a. User
   - UserId -> User_type

b. Admin
   - AdminId -> name
   - AdminId -> password

c. Member
   - MemberID -> Name
   - MemberID -> Email
   - MemberID -> password
   - MemberID -> phoneNumber
   - MemberID -> address

d. Buyer
   - BuyerID -> shippingAddress
   - BuyerID -> creditCardNum

e. Seller
   - SellerID -> bankAccountNum
   - SellerID -> routingNumber

f. Bids
   - {BuyerID,SellerID,ItemID} -> bidStatus

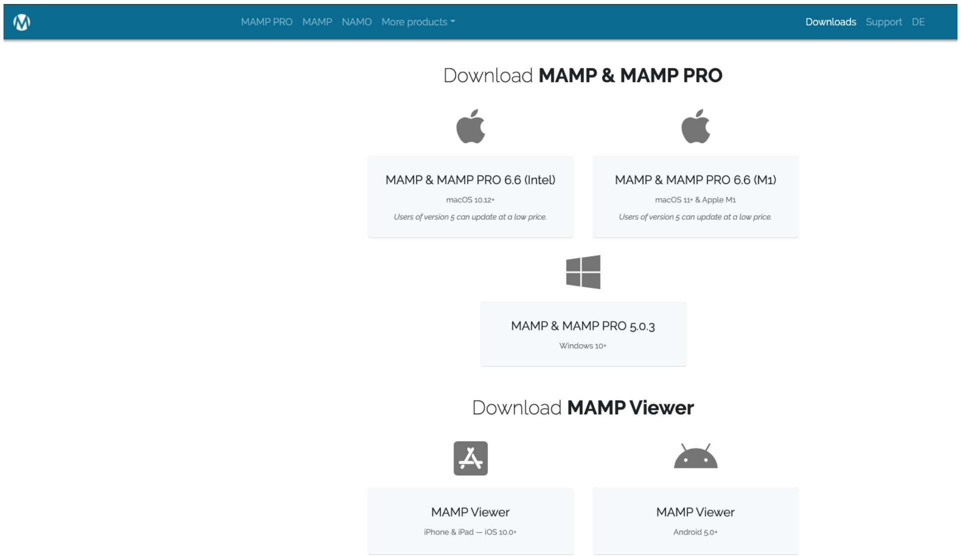
g. Items
   - ItemID -> Item_type

h. Feedback
   - {reviewerID,revieweeID} -> feedbackMessage

Database System:

Installing and invoking the system can be accomplished by installing the application MAMP. By designating the correct document root and importing the database file into the web application in the description below, the application for the project can be accessed in the browser at ‘localhost:PORT_NUMBER/’, where PORT_NUMBER is the designated port number in your setup.

Install MAMP from https://www.mamp.info/en/downloads/



Download **MAMP & MAMP PRO**

MAMP & MAMP PRO 6.6 (Intel)
macOS 10.12+
Users of version 5 can update at a low price.

MAMP & MAMP PRO 6.6 (M1)
macOS 11+ & Apple M1
Users of version 5 can update at a low price.

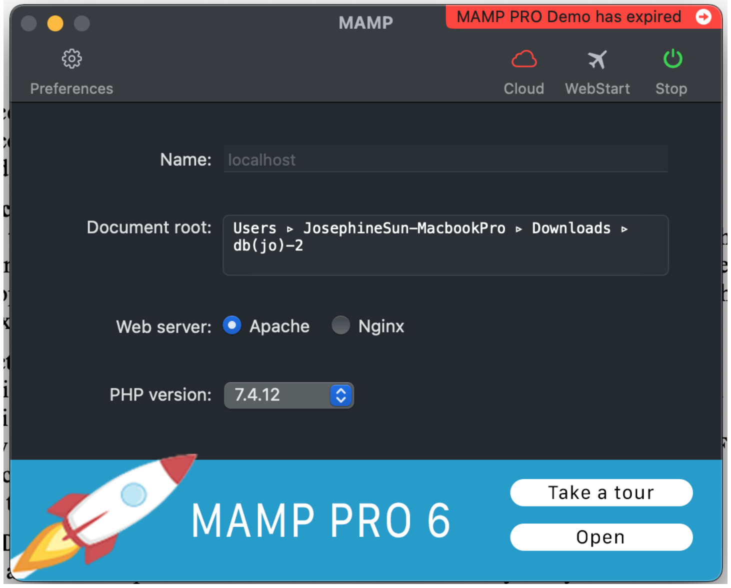
MAMP & MAMP PRO 5.0.3
Windows 10+

Download **MAMP Viewer**

MAMP Viewer
iPhone & iPad — iOS 10.0+

MAMP Viewer
Android 5.0+

Open the MAMP Application and select ‘Preferences’.



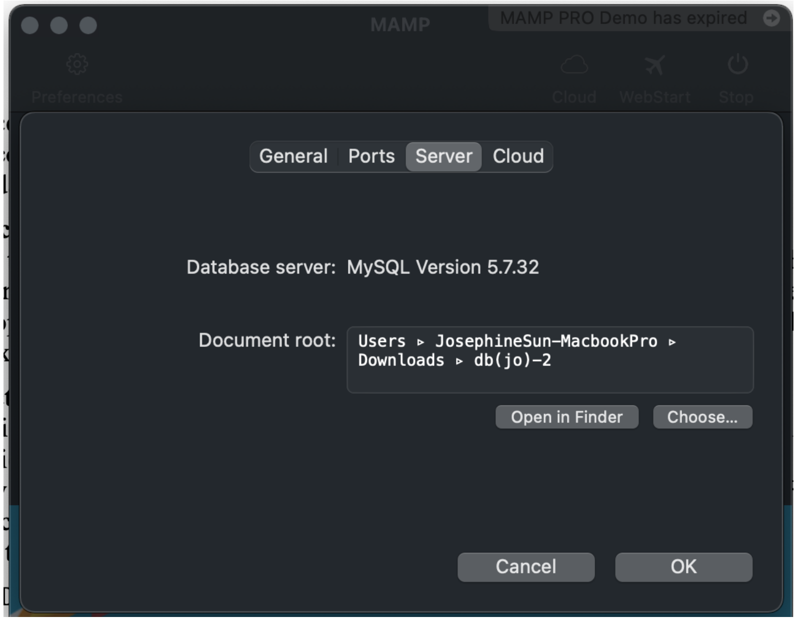
Name: localhost

Document root: Users ▸ JosephineSun-MacbookPro ▸ Downloads ▸ db(jo)-2

Web server: ● Apache ○ Nginx

PHP version: 7.4.12

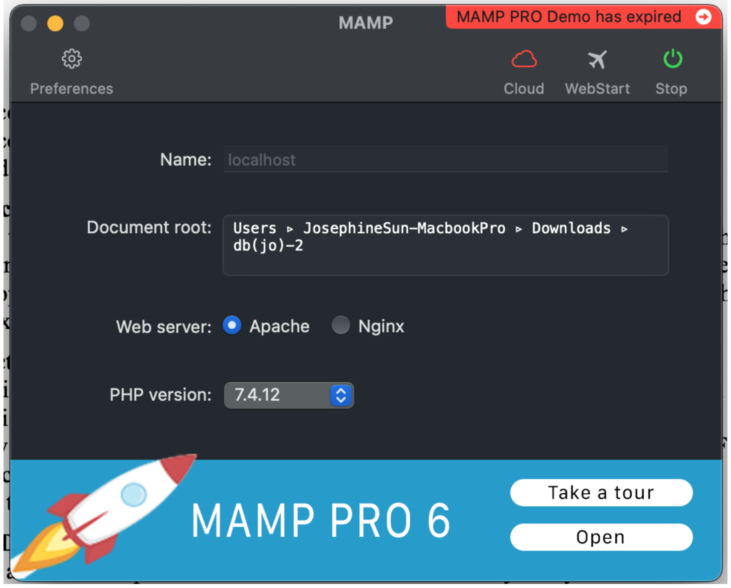
Navigate to ‘Server’ and set the ‘Document root’ to the correct folder.



Database server: MySQL Version 5.7.32

Document root: Users ▸ JosephineSun-MacbookPro ▸ Downloads ▸ db(jo)-2

Click ‘Start’ and then click ‘Webstart’ to open up in the browser.



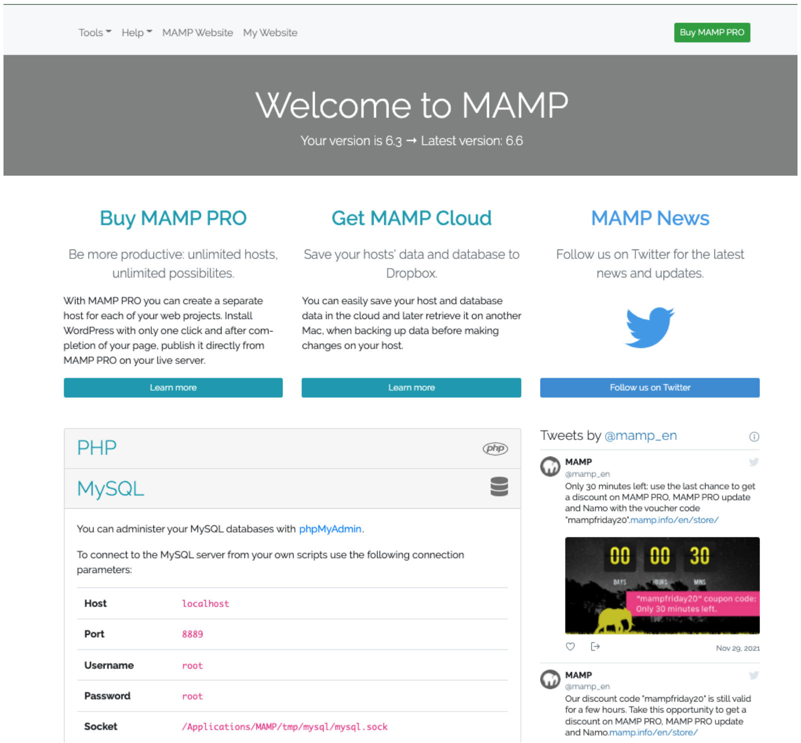
Name: localhost

Document root: Users ▸ JosephineSun-MacbookPro ▸ Downloads ▸ db(jo)-2

Web server: ● Apache ○ Nginx

PHP version: 7.4.12

Navigate down to 'MySQL' and click the link 'phpMyAdmin' to set up the database.



Welcome to MAMP

Your version is 6.3 → Latest version: 6.6

Buy MAMP PRO

Be more productive: unlimited hosts, unlimited possibilites.

With MAMP PRO you can create a separate host for each of your web projects. Install WordPress with only one click and after completion of your page, publish it directly from MAMP PRO on your live server.

Get MAMP Cloud

Save your hosts' data and database to Dropbox.

You can easily save your host and database data in the cloud and later retrieve it on another Mac, when backing up data before making changes on your host.

MAMP News

Follow us on Twitter for the latest news and updates.

You can administer your MySQL databases with phpMyAdmin.

To connect to the MySQL server from your own scripts use the following connection parameters:

Host	localhost
Port	8889
Username	root
Password	root
Socket	/Applications/MAMP/tmp/mysql/mysql.sock

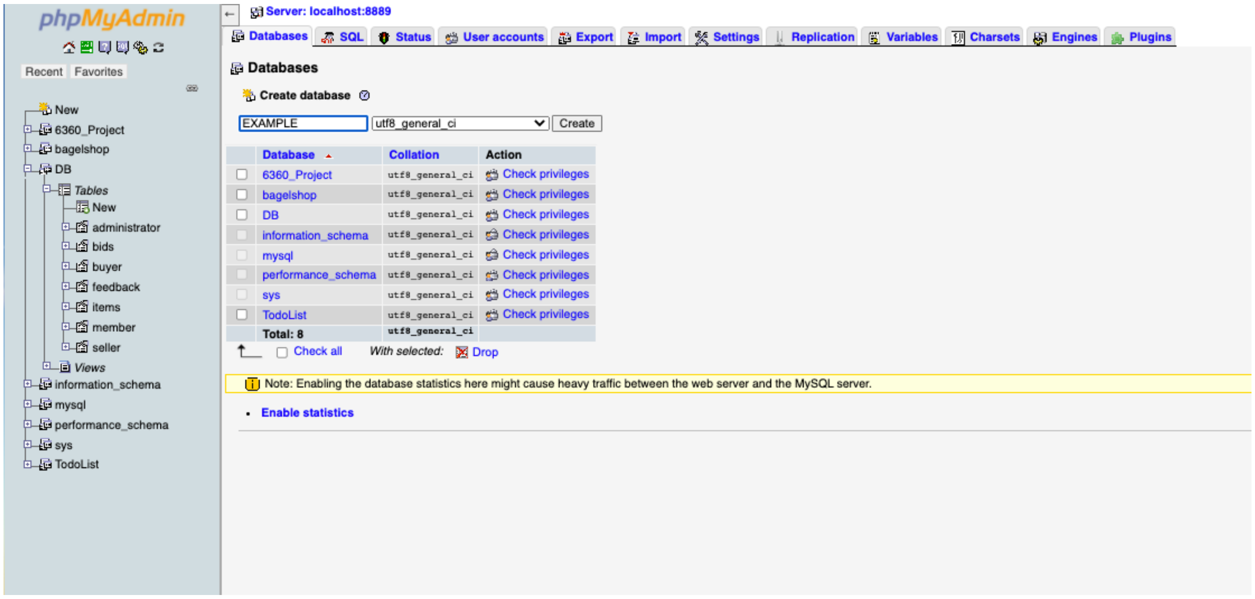
Tweets by @mamp_en

MAMP
@mamp_en
Only 30 minutes left: use the last chance to get a discount on MAMP PRO, MAMP PRO update and Namo with the voucher code "mampfriday20".mamp.info/en/store/

Nov 29, 2021

MAMP
@mamp_en
Our discount code "mampfriday20" is still valid for a few hours. Take this opportunity to get a discount on MAMP PRO, MAMP PRO update and Namo.mamp.info/en/store/

On the left, select 'New' to create a new database. In the example below, the database it titled 'EXAMPLE'.



Server: localhost:8889

Databases

Create database

EXAMPLE | utf8_general_ci

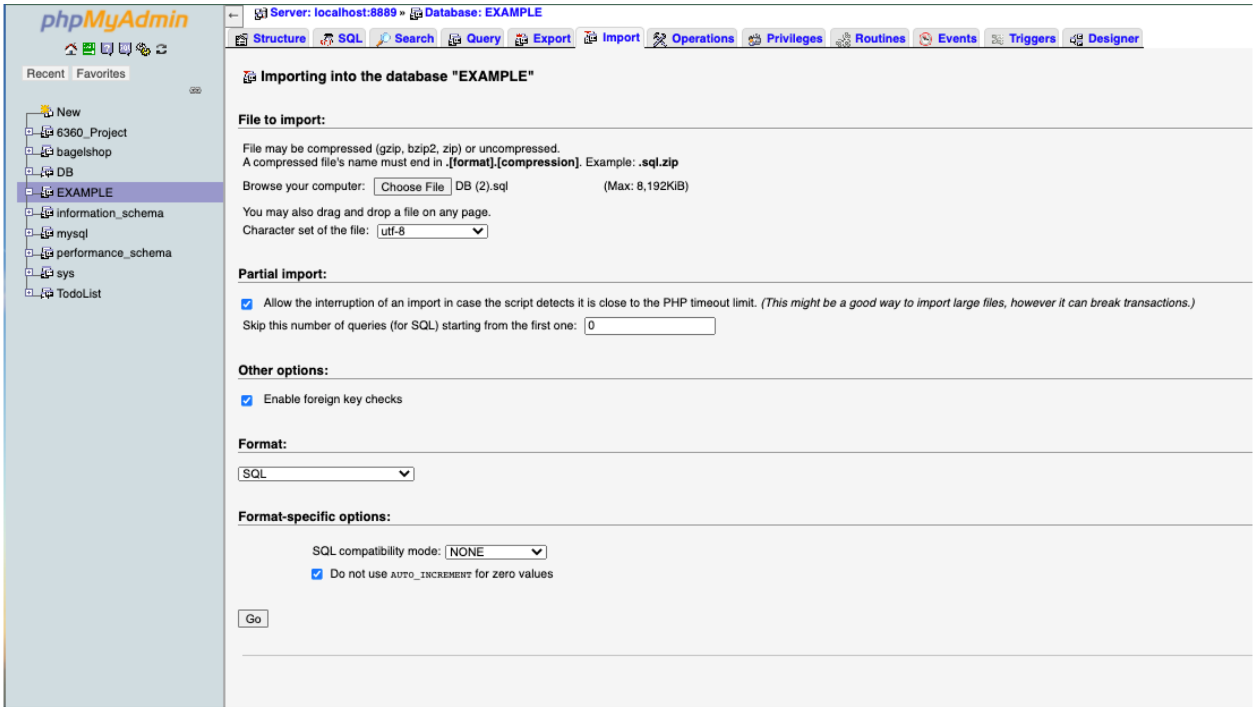
	Database	Collation	Action
□	6360_Project	utf8_general_ci	Check privileges
□	bagelshop	utf8_general_ci	Check privileges
□	DB	utf8_general_ci	Check privileges
□	information_schema	utf8_general_ci	Check privileges
□	mysql	utf8_general_ci	Check privileges
□	performance_schema	utf8_general_ci	Check privileges
□	sys	utf8_general_ci	Check privileges
□	TodoList	utf8_general_ci	Check privileges
	Total: 8	utf8_general_ci	

□ Check all With selected: Drop

Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.

- Enable statistics

Navigate the top tabs to ‘import’ the database file.



Importing into the database "EXAMPLE"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

Browse your computer: Choose File DB (2).sql (Max: 8,192KiB)

You may also drag and drop a file on any page.

Character set of the file: utf-8

Partial import:

☑ Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large files, however it can break transactions.)*

Skip this number of queries (for SQL) starting from the first one: 0

Other options:

☑ Enable foreign key checks

Format:

SQL

Format-specific options:

SQL compatibility mode: NONE

☑ Do not use AUTO_INCREMENT for zero values

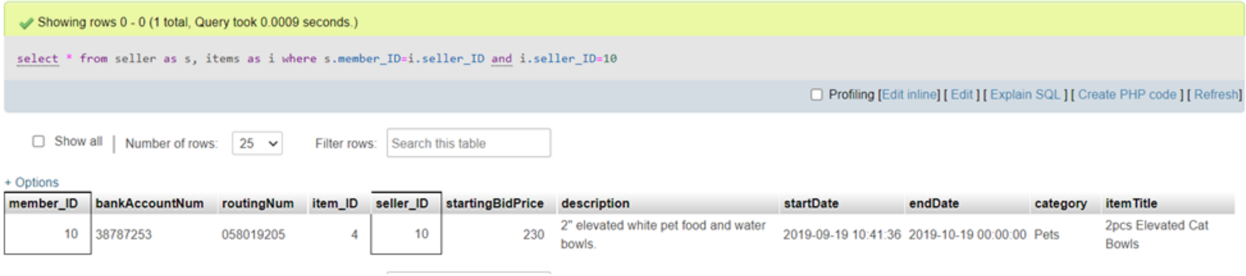
Additional Queries and Views

Query

Seller Table:

Retrieve items listed under a specified seller, such as where the seller_ID is 10.

SELECT * FROM seller as s, items as i WHERE s.member_ID=i.seller_ID AND i.seller_ID=10;



✓ Showing rows 0 - 0 (1 total, Query took 0.0009 seconds.)

```
select * from seller as s, items as i where s.member_ID=i.seller_ID and i.seller_ID=10
```

□ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

□ Show all | Number of rows: 25 Filter rows: Search this table

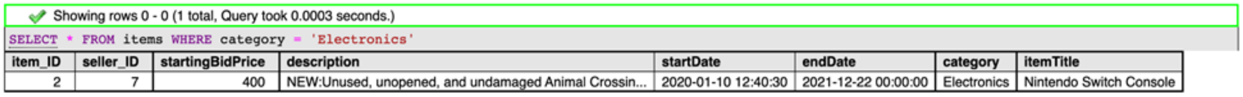
+ Options

member_ID	bankAccountNum	routingNum	item_ID	seller_ID	startingBidPrice	description	startDate	endDate	category	itemTitle
10	38787253	058019205	4	10	230	2" elevated white pet food and water bowls.	2019-09-19 10:41:36	2019-10-19 00:00:00	Pets	2pcs Elevated Cat Bowls

Items Table:

Retrieve the set of items under a specified category, such as “Electronics”

SELECT * FROM items WHERE category = ‘Electronics’;



✓ Showing rows 0 - 0 (1 total, Query took 0.0003 seconds.)

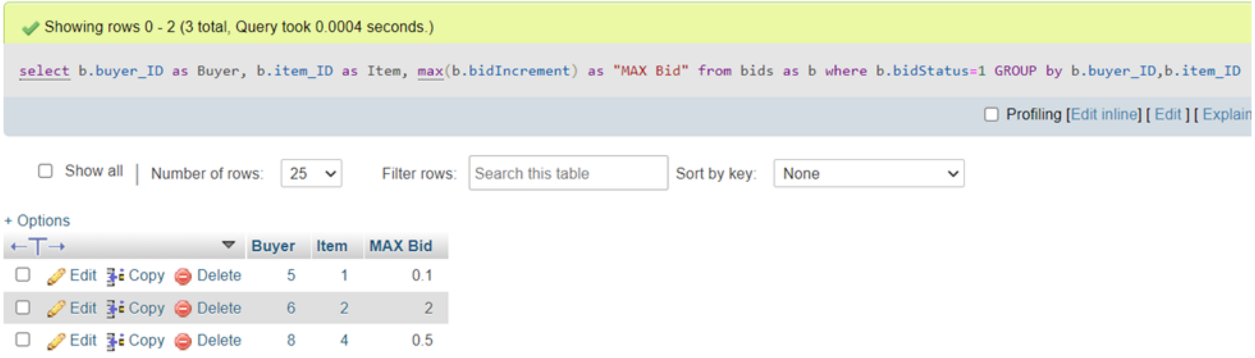
```
SELECT * FROM items WHERE category = 'Electronics'
```

item_ID	seller_ID	startingBidPrice	description	startDate	endDate	category	itemTitle
2	7	400	NEW:Unused, unopened, and undamaged Animal Crossin...	2020-01-10 12:40:30	2021-12-22 00:00:00	Electronics	Nintendo Switch Console

Bids Table:

Retrieve the max bid increment placed by each buyer for each item.

SELECT b.buyer_ID as Buyer, b.item_ID as Item, max(b.bidInrement) as “MAX Bid” from bids as b where b.bidStatus group by b.buyer_ID, b.item_ID;



✓ Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
select b.buyer_ID as Buyer, b.item_ID as Item, max(b.bidIncrement) as "MAX Bid" from bids as b where b.bidStatus=1 GROUP by b.buyer_ID,b.item_ID
```

□ Profiling [Edit inline] [Edit] [Explain

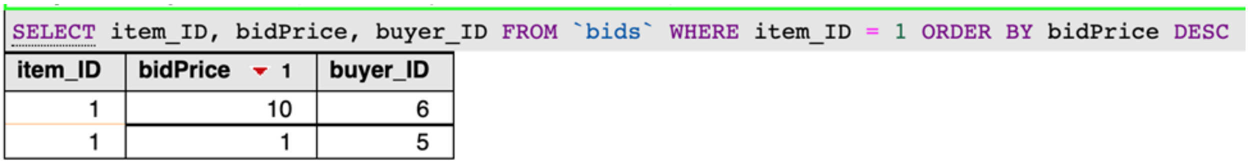
□ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

←T→				Buyer	Item	MAX Bid
□	Edit	Copy	Delete	5	1	0.1
□	Edit	Copy	Delete	6	2	2
□	Edit	Copy	Delete	8	4	0.5

Retrieve the bids placed on a specific item (in this case the item whose id is 1) and order the bids made in descending order.

SELECT item_ID, bidPrice, buyer_ID FROM `bids` WHERE item_ID = 1 ORDER BY bidPrice DESC:



```
SELECT item_ID, bidPrice, buyer_ID FROM `bids` WHERE item_ID = 1 ORDER BY bidPrice DESC
```

item_ID	bidPrice ▼ 1	buyer_ID
1	10	6
1	1	5

Views

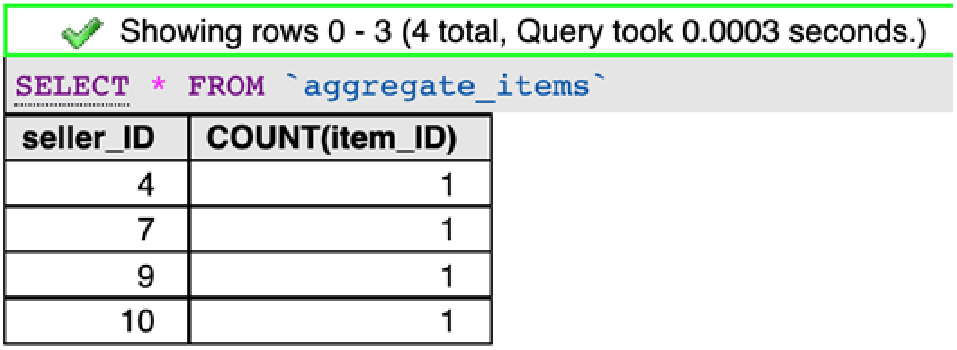
Items Table:

View for the number of items listed by each seller:

Create view aggregate_items AS

(Select sellerid, count(item_id) from items

Group by seller_id);



✓ Showing rows 0 - 3 (4 total, Query took 0.0003 seconds.)

```
SELECT * FROM `aggregate_items`
```

seller_ID	COUNT(item_ID)
4	1
7	1
9	1
10	1

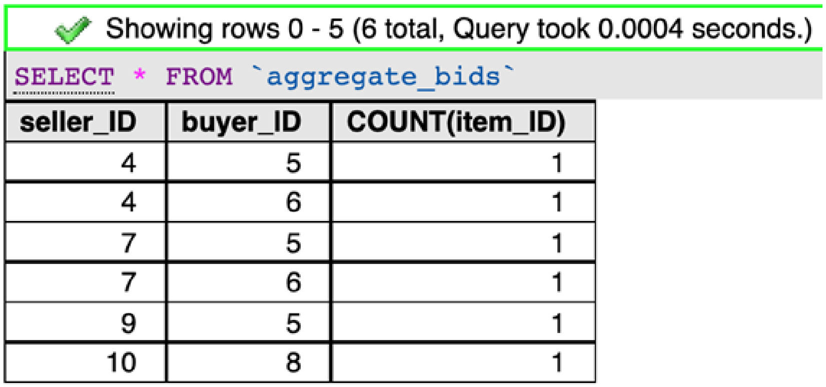
Bids Table:

View for the number of items of each seller that each buyer has bid for:

Create view aggregate_bids AS

(Select seller_id, buyer_id, count(item_ID) from bids

Group by seller_id, buyer_id);



✓ Showing rows 0 - 5 (6 total, Query took 0.0004 seconds.)

```
SELECT * FROM `aggregate_bids`
```

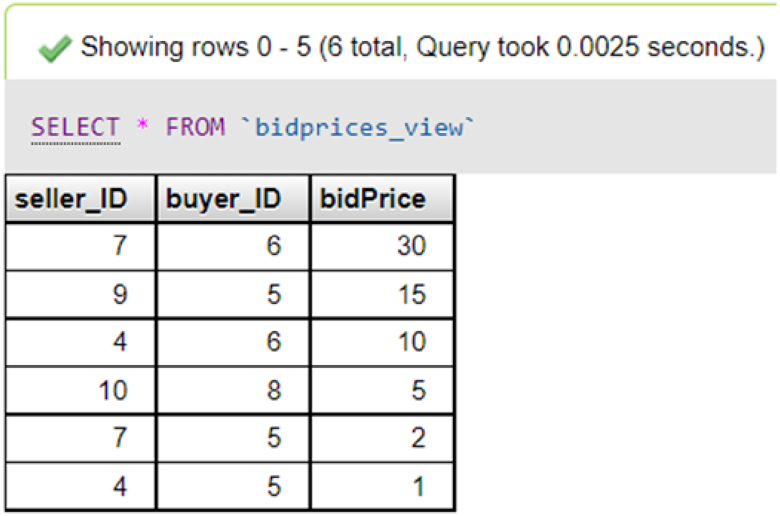
seller_ID	buyer_ID	COUNT(item_ID)
4	5	1
4	6	1
7	5	1
7	6	1
9	5	1
10	8	1

View for displaying the bids from the highest to the lowest bid price:

Create view aggregate_bids AS

(Select seller_id, buyer_id, bidPrice from bids

Order By bidPrice DESC);



✓ Showing rows 0 - 5 (6 total, Query took 0.0025 seconds.)

```
SELECT * FROM `bidprices_view`
```

seller_ID	buyer_ID	bidPrice
7	6	30
9	5	15
4	6	10
10	8	5
7	5	2
4	5	1

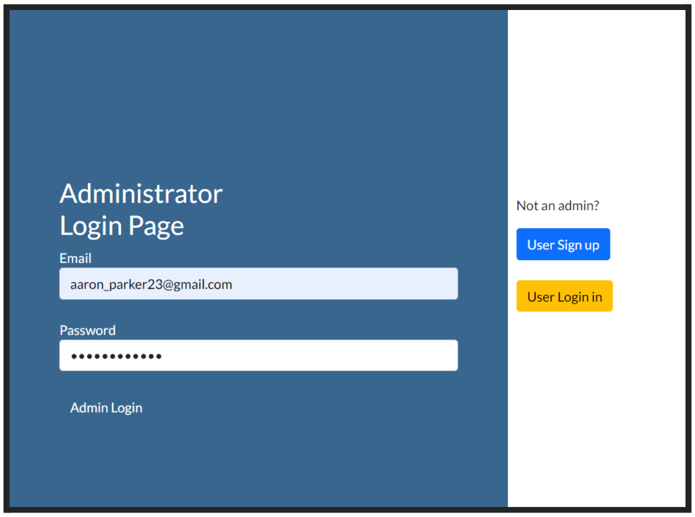
User Application Interface

The system user interface is built using HTML and php, which can be run locally from the browser. Given that the database is set up, one can begin the application by navigating to the 'main.php' file in their browser. The applied functions and their respective user interface design are listed below. The functions are implemented as SQL queries in the php and HTML code and the results are presented in the interface, as in the below applications.

Admin Functions/Operations:

Admin Login page:

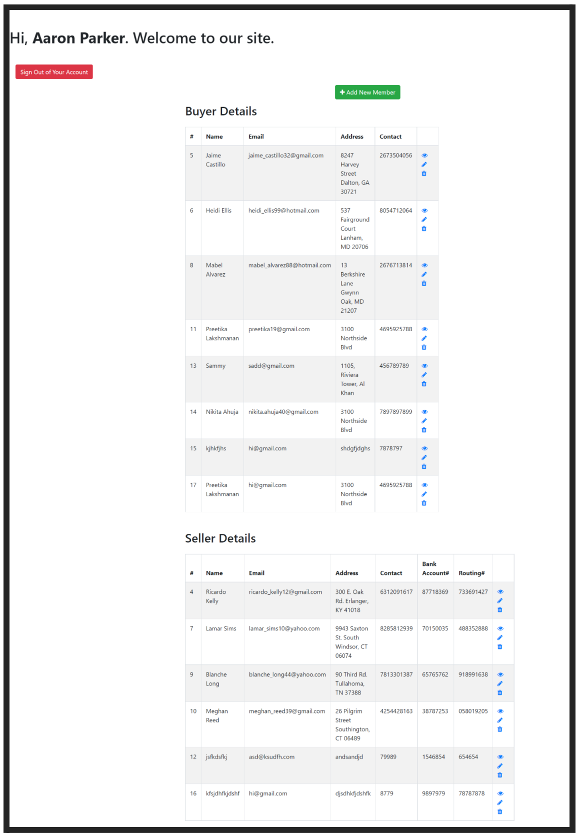
The admin login info is submitted and verified against the login info for administrator accounts in the database.



Email: aaron_parker23@gmail.com

Password: ••••••••••••

Admin welcome page:
Each user’s information is returned using a select query to return corresponding information of all Buyer and Seller member accounts.



Hi, **Aaron Parker**. Welcome to our site.

Buyer Details

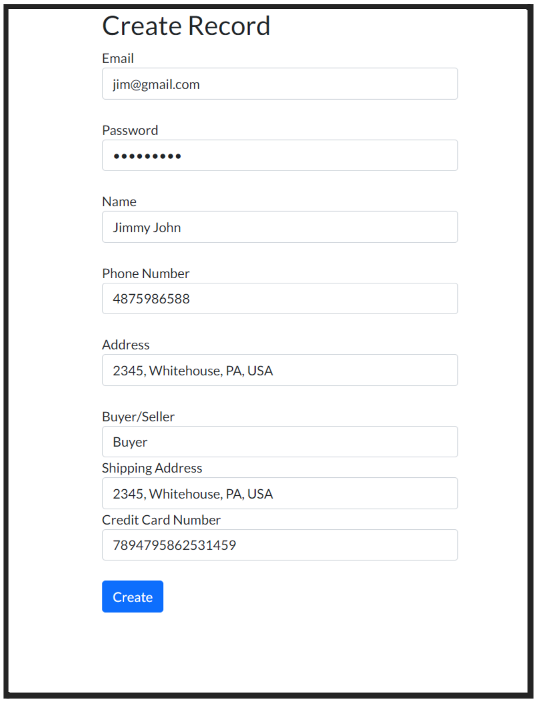
#	Name	Email	Address	Contact	
5	Jaime Castillo	jaime_castillo32@gmail.com	8247 Harvey Street Dalton, GA 30721	2673504056	👁 ✎ 🗑
6	Heidi Ellis	heidi_ellis99@hotmail.com	537 Fairground Court Lanham, MD 20706	8054712064	👁 ✎ 🗑
8	Mabel Alvarez	mabel_alvarez88@hotmail.com	13 Berkshire Lane Gwynn Oak, MD 21207	2676713814	👁 ✎ 🗑
11	Preetika Lakshmanan	preetika19@gmail.com	3100 Northside Blvd	4695925788	👁 ✎ 🗑
13	Sammy	sadd@gmail.com	1105, Riviera Tower, Al Khan	456789789	👁 ✎ 🗑
14	Nikita Ahuja	nikita.ahuja40@gmail.com	3100 Northside Blvd	7897897899	👁 ✎ 🗑
15	kjhkfjhs	hi@gmail.com	shdgfjdghs	7878797	👁 ✎ 🗑
17	Preetika Lakshmanan	hi@gmail.com	3100 Northside Blvd	4695925788	👁 ✎ 🗑

Seller Details

#	Name	Email	Address	Contact	Bank Account#	Routing#	
4	Ricardo Kelly	ricardo_kelly12@gmail.com	300 E. Oak Rd. Erlanger, KY 41018	6312091617	87718369	733691427	👁 ✎ 🗑
7	Lamar Sims	lamar_sims10@yahoo.com	9943 Saxton St. South Windsor, CT 06074	8285812939	70150035	488352888	👁 ✎ 🗑
9	Blanche Long	blanche_long44@yahoo.com	90 Third Rd. Tullahoma, TN 37388	7813301387	65765762	918991638	👁 ✎ 🗑
10	Meghan Reed	meghan_reed39@gmail.com	26 Pilgrim Street Southington, CT 06489	4254428163	38787253	058019205	👁 ✎ 🗑
12	jsfkdsfkj	asd@ksudfh.com	andsandjd	79989	1546854	654654	👁 ✎ 🗑
16	kfsjdhfkjdshf	hi@gmail.com	djsdhkfjdshfk	8779	9897979	78787878	👁 ✎ 🗑

Admin create a new member page:

Use of Insert query in the database to add new members



Create Record

Email

jim@gmail.com

Password

•••••••••

Name

Jimmy John

Phone Number

4875986588

Address

2345, Whitehouse, PA, USA

Buyer/Seller

Buyer

Shipping Address

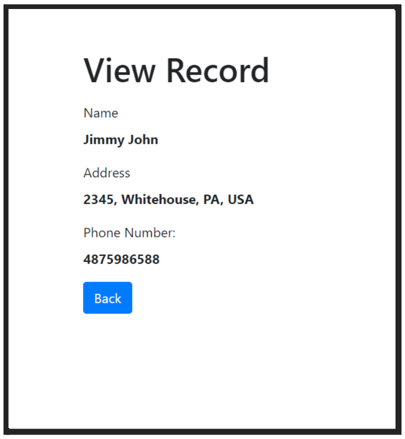
2345, Whitehouse, PA, USA

Credit Card Number

7894795862531459

Admin view a member page:

Use of Select query to present a specific member’s information.



View Record

Name

Jimmy John

Address

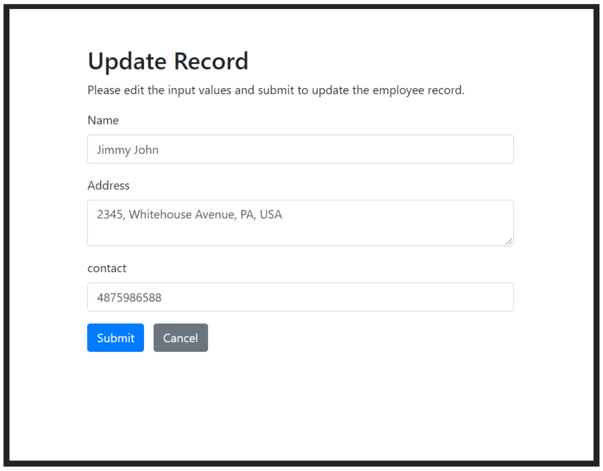
2345, Whitehouse, PA, USA

Phone Number:

4875986588

Admin Update a member page:

Update query to the members table.



Update Record

Please edit the input values and submit to update the employee record.

Name

Jimmy John

Address

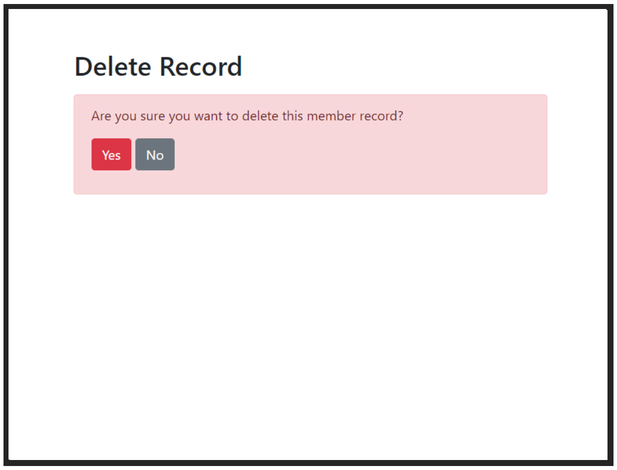
2345, Whitehouse Avenue, PA, USA

contact

4875986588

Admin delete a member page:

Delete query to a specific tuple in the member's table.



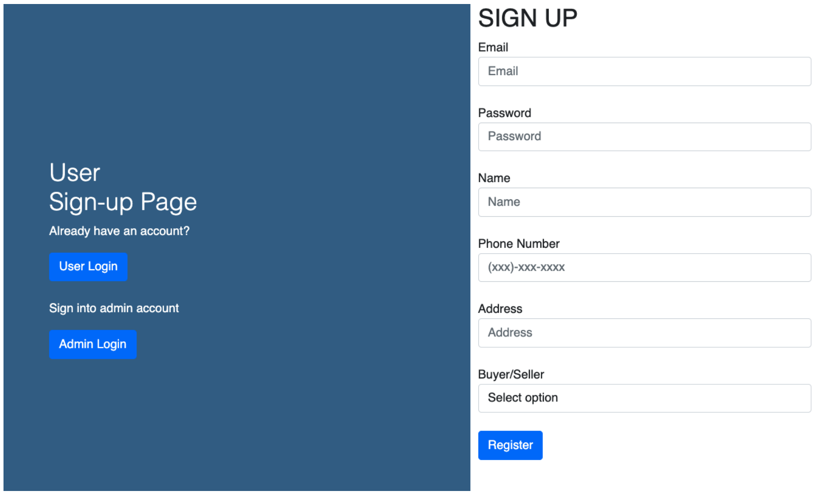
Delete Record

Are you sure you want to delete this member record?

Seller Functions/Operations:

User signup page

Insert query to the member's table and the seller/buyer table depending on the account type selected by the user.



User
Sign-up Page

Already have an account?

User Login

Sign into admin account

Admin Login

SIGN UP

Email
Email

Password
Password

Name
Name

Phone Number
(xxx)-xxx-xxxx

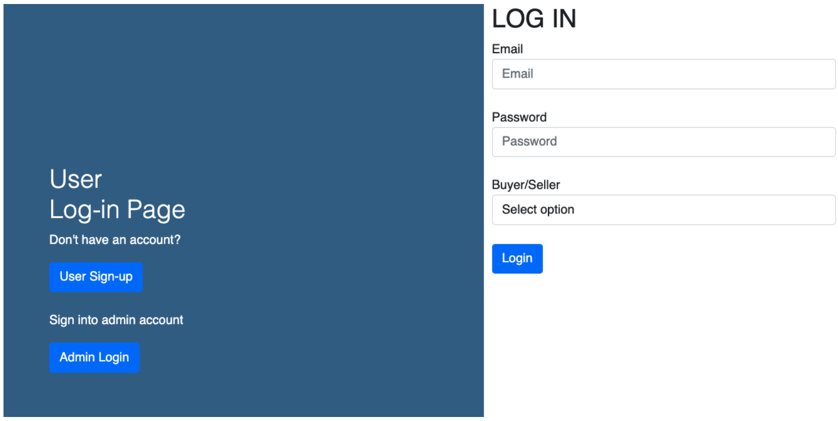
Address
Address

Buyer/Seller
Select option

Register

User login page:

User information is validated against all the login information for buyer/sellers to see if the login information is valid.



User
Log-in Page

Don't have an account?

User Sign-up

Sign into admin account

Admin Login

LOG IN

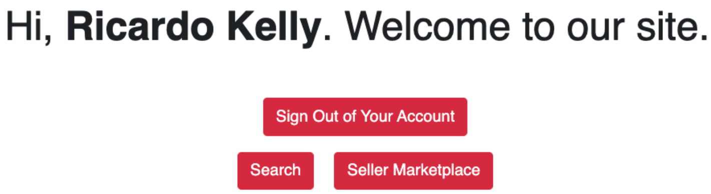
Email
Email

Password
Password

Buyer/Seller
Select option

Login

Seller Homepage:



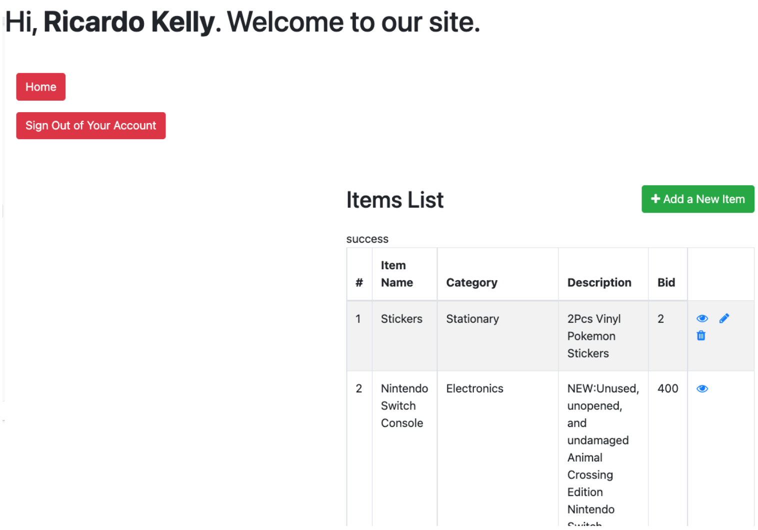
Hi, **Ricardo Kelly**. Welcome to our site.

Sign Out of Your Account

Search

Seller Marketplace

Seller Marketplace:
Select query returning all the existing items, and allowing modifications only to items created by the seller through validation in php.



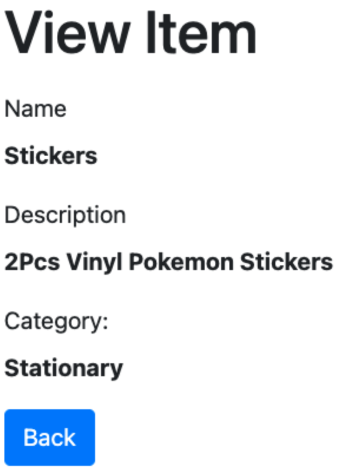
Hi, **Ricardo Kelly**. Welcome to our site.

Items List

success

#	Item Name	Category	Description	Bid	
1	Stickers	Stationary	2Pcs Vinyl Pokemon Stickers	2	👁 ✎ 🗑
2	Nintendo Switch Console	Electronics	NEW:Unused, unopened, and undamaged Animal Crossing Edition Nintendo Switch	400	👁

View Items:
Select query returning information of a specific item.



View Item

Name

Stickers

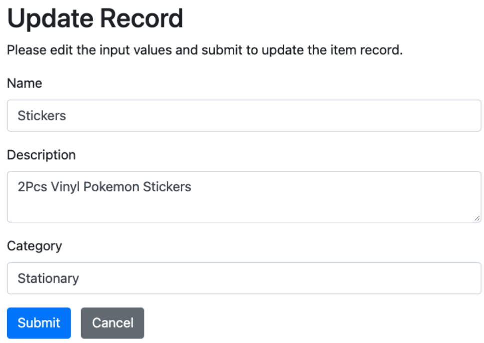
Description

2Pcs Vinyl Pokemon Stickers

Category:

Stationary

Edit seller’s items:
Update query to the items table given a specific item id.



Update Record

Please edit the input values and submit to update the item record.

Name

Stickers

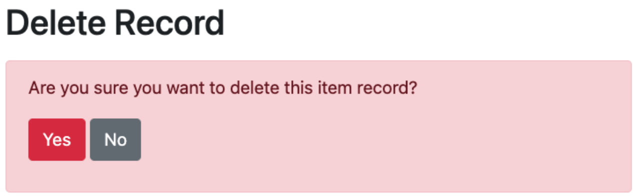
Description

2Pcs Vinyl Pokemon Stickers

Category

Stationary

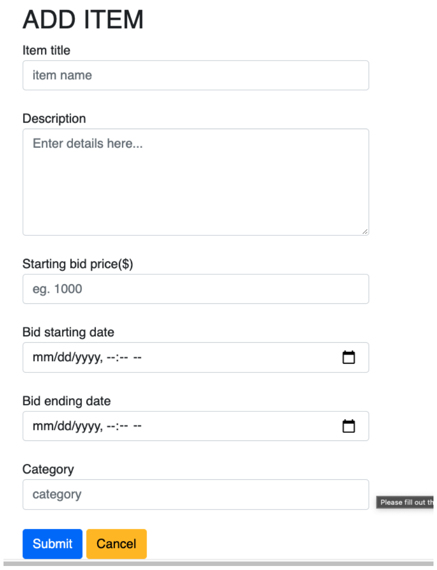
Delete Seller’s items:
Delete query to the items table given a specific item id.



Delete Record

Are you sure you want to delete this item record?

Seller add item:
Insert query to the items table given the user’s input and seller id. Form fields are designed to be required.



ADD ITEM

Item title

item name

Description

Enter details here...

Starting bid price($)

eg. 1000

Bid starting date

mm/dd/yyyy, --:-- --

Bid ending date

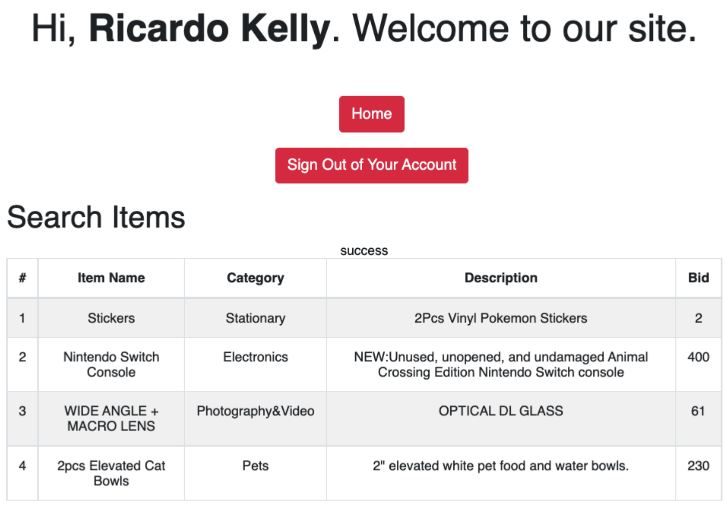
mm/dd/yyyy, --:-- --

Category

category

Please fill out th

Seller Search items:
Select query returning a list of items to the user.



Hi, **Ricardo Kelly**. Welcome to our site.

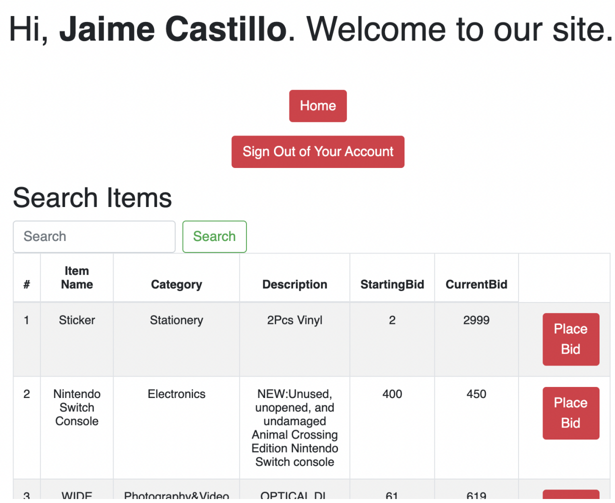
Search Items

success

#	Item Name	Category	Description	Bid
1	Stickers	Stationary	2Pcs Vinyl Pokemon Stickers	2
2	Nintendo Switch Console	Electronics	NEW:Unused, unopened, and undamaged Animal Crossing Edition Nintendo Switch console	400
3	WIDE ANGLE + MACRO LENS	Photography&Video	OPTICAL DL GLASS	61
4	2pcs Elevated Cat Bowls	Pets	2" elevated white pet food and water bowls.	230

Buyer Functions/Operations:

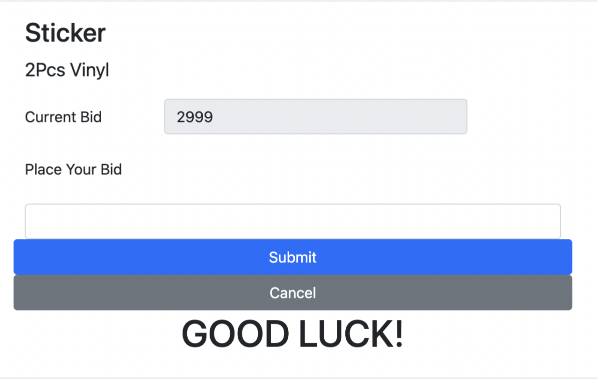
Buyer can search items and place a bid for each item.
Select query returning a list of items to the user.



Hi, **Jaime Castillo**. Welcome to our site.

Search Items

#	Item Name	Category	Description	StartingBid	CurrentBid	
1	Sticker	Stationery	2Pcs Vinyl	2	2999	Place Bid
2	Nintendo Switch Console	Electronics	NEW:Unused, unopened, and undamaged Animal Crossing Edition Nintendo Switch console	400	450	Place Bid
3	WIDE	Photography&Video	OPTICAL DL	61	619	



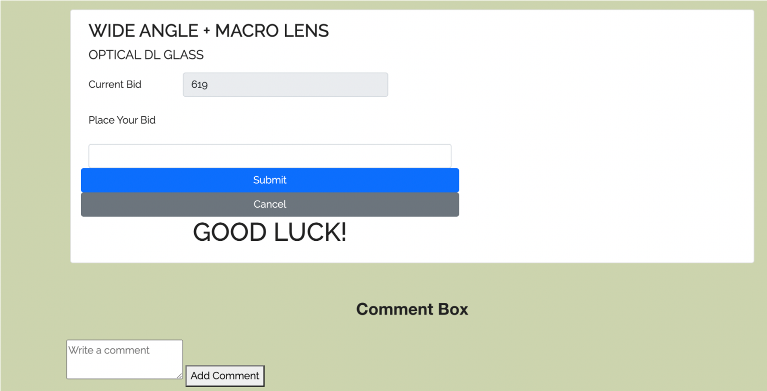
Sticker

2Pcs Vinyl

Current Bid 2999

Place Your Bid

Buyer can give feedback,
Another Insert query to the feedback table.



WIDE ANGLE + MACRO LENS

OPTICAL DL GLASS

Current Bid 619

Place Your Bid

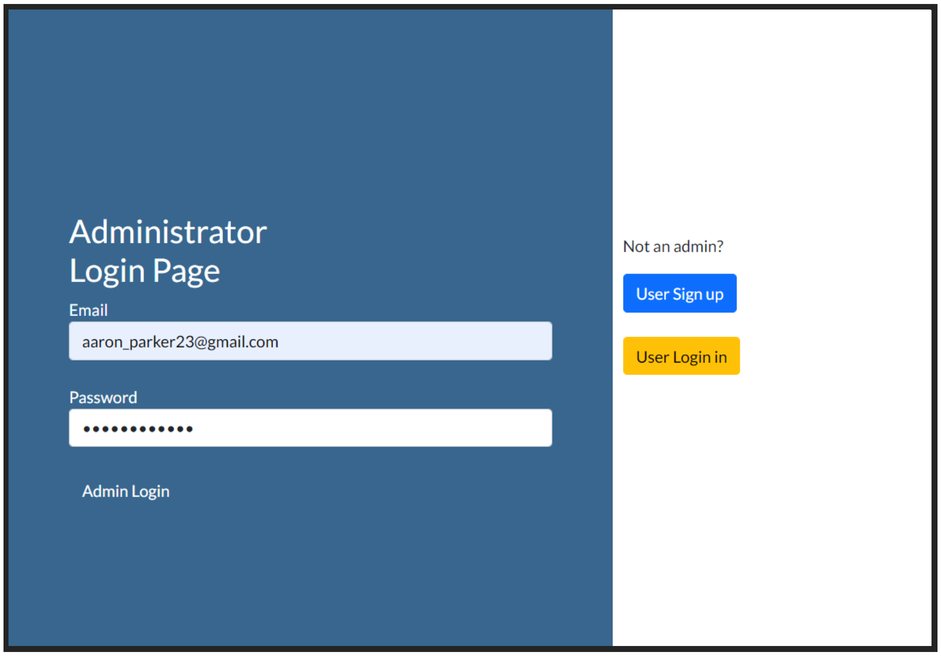
a) A list of alternative options is offered to the user.

Starting from main.php, the user is prompted for an

> administrator login

> user sign up

> user login.



Administrator Login Page

Email: aaron_parker23@gmail.com

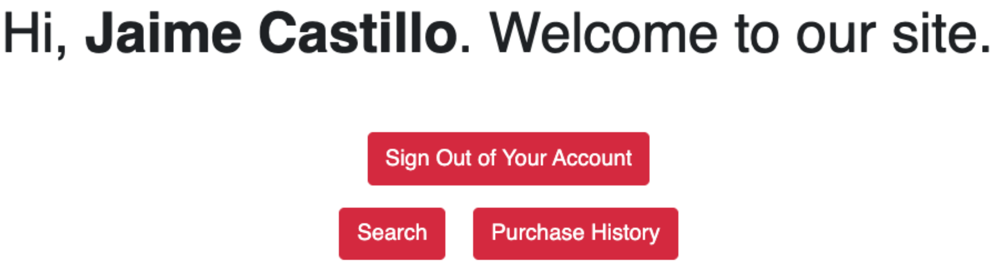
Password: ••••••••••••

From the user homepage, welcome.php, the user may

> Sign out

> Search items

> View their purchase history



Hi, **Jaime Castillo**. Welcome to our site.

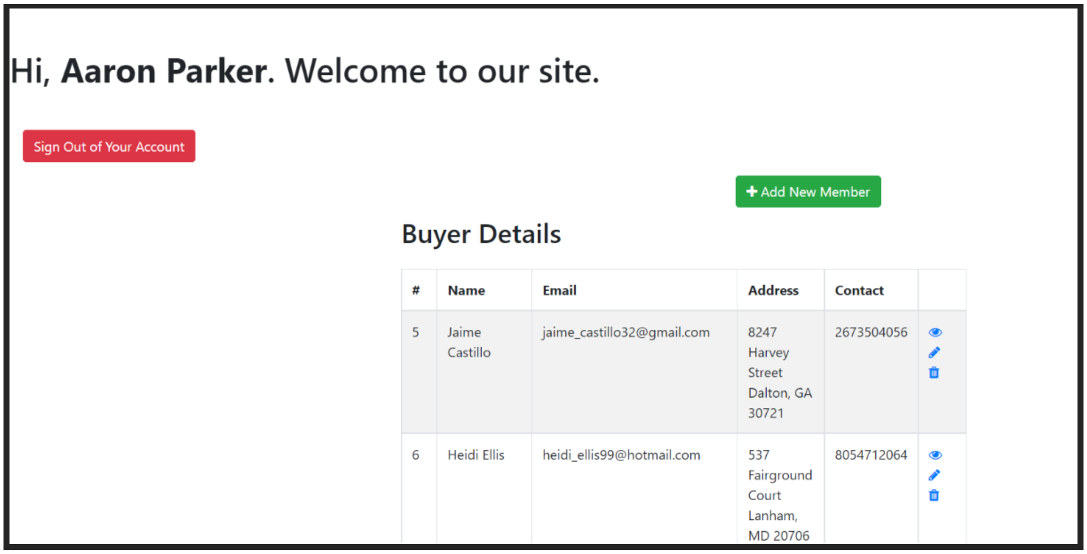
b) The user selects an alternative.

Once signed in as an administrator, the user is given the option to

> Sign out

> Add a new member (buyer/seller)

> view/edit/delete user accounts. In the case below, the admin is ‘aaron parker’.



Hi, **Aaron Parker**. Welcome to our site.

Buyer Details

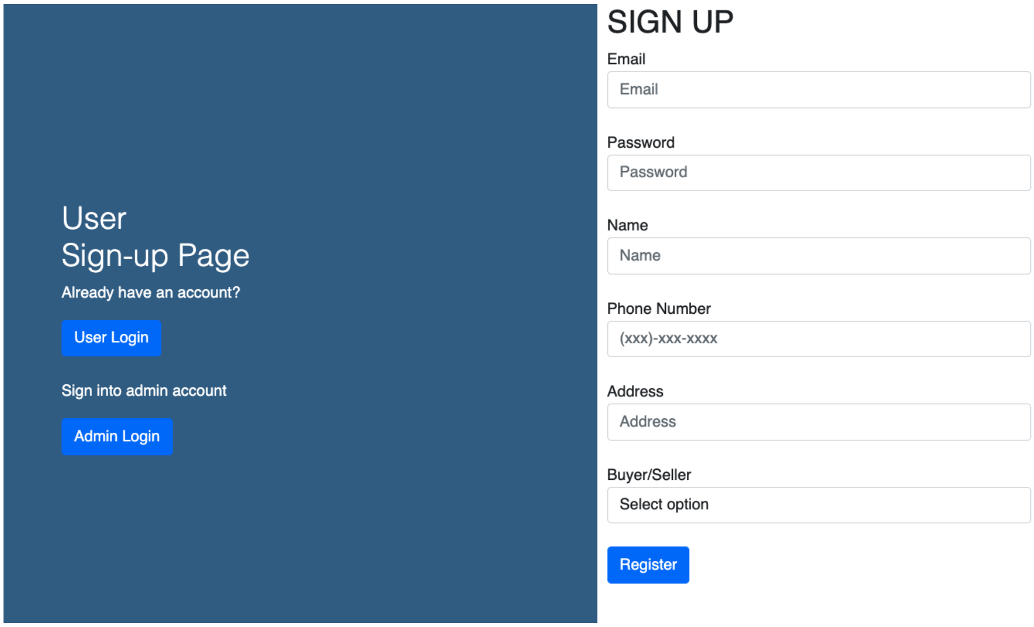
#	Name	Email	Address	Contact	
5	Jaime Castillo	jaime_castillo32@gmail.com	8247 Harvey Street Dalton, GA 30721	2673504056	👁 ✎ 🗑
6	Heidi Ellis	heidi_ellis99@hotmail.com	537 Fairground Court Lanham, MD 20706	8054712064	👁 ✎ 🗑

Instead of signing in as an admin, the user may also opt to sign up for an account. Here, the user is given the options to

> Re-direct to user login

> Re-direct to admin login

> Fill the form to register an account



User
Sign-up Page

Already have an account?

Sign into admin account

SIGN UP

Email
Email

Password
Password

Name
Name

Phone Number
(xxx)-xxx-xxxx

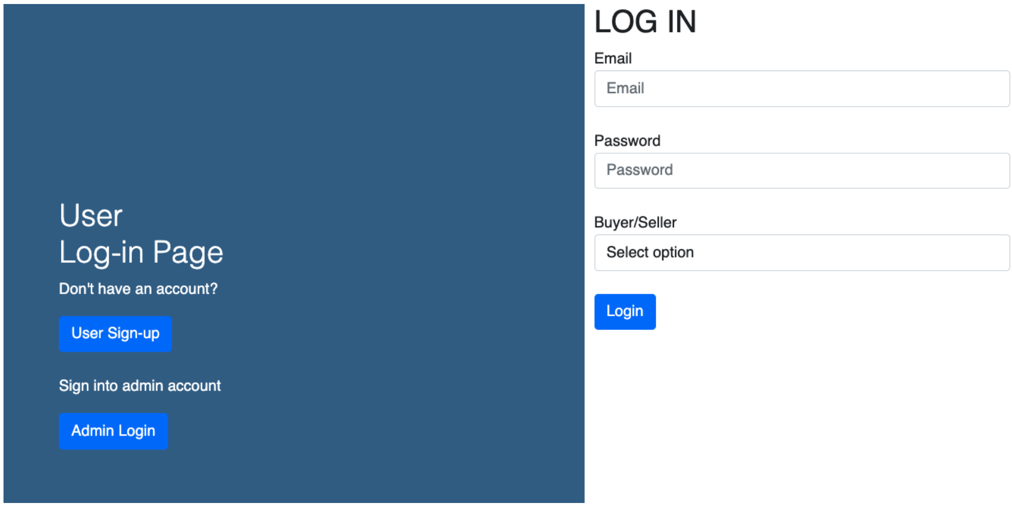
Address
Address

Buyer/Seller
Select option

If the user initially opts to login to a user account, the options are similar that the user may

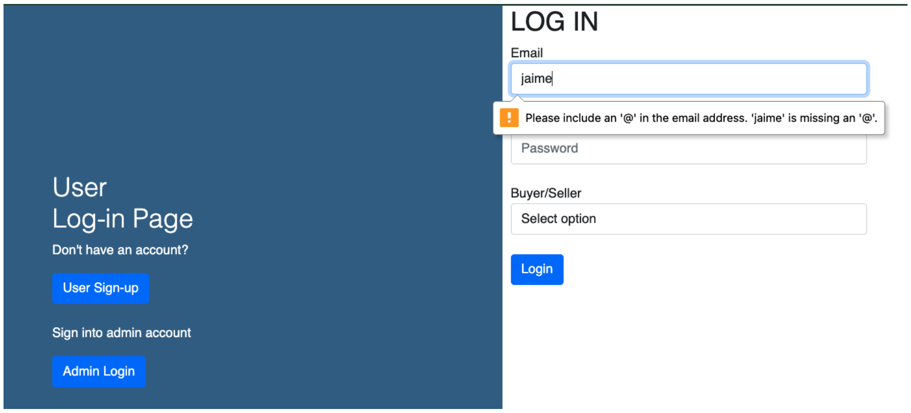
> Re-direct to user sign up

> Re-direct to admin login
> Fill the form to login to their account.



c) The system prompts the user for appropriate input values.

In the various forms, the application also prompts the user for appropriate input values, such as in the user login screen.

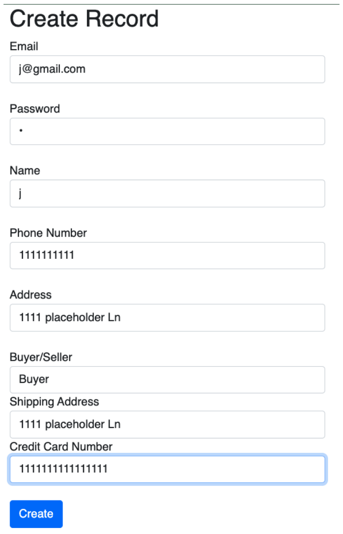


Email: jaime

Please include an '@' in the email address. 'jaime' is missing an '@'.

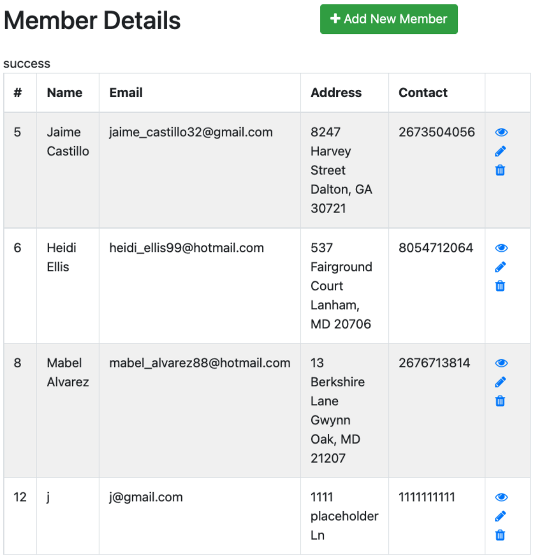
d) The system accesses the database to perform the appropriate queries and/or modifications.

In the admin homepage, the admin may add new users, which redirects to the following page and inserts another member into the database, which is reflected on the page.



Create Record

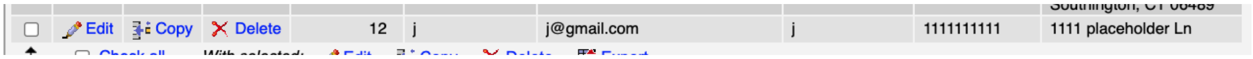
Email	j@gmail.com
Password	•
Name	j
Phone Number	1111111111
Address	1111 placeholder Ln
Buyer/Seller	Buyer
Shipping Address	1111 placeholder Ln
Credit Card Number	1111111111111111



Member Details

success

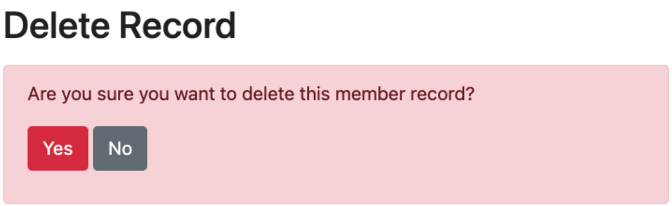
#	Name	Email	Address	Contact	
5	Jaime Castillo	jaime_castillo32@gmail.com	8247 Harvey Street Dalton, GA 30721	2673504056	👁 ✎ 🗑
6	Heidi Ellis	heidi_ellis99@hotmail.com	537 Fairground Court Lanham, MD 20706	8054712064	👁 ✎ 🗑
8	Mabel Alvarez	mabel_alvarez88@hotmail.com	13 Berkshire Lane Gwynn Oak, MD 21207	2676713814	👁 ✎ 🗑
12	j	j@gmail.com	1111 placeholder Ln	1111111111	👁 ✎ 🗑



Southington, CT 06489

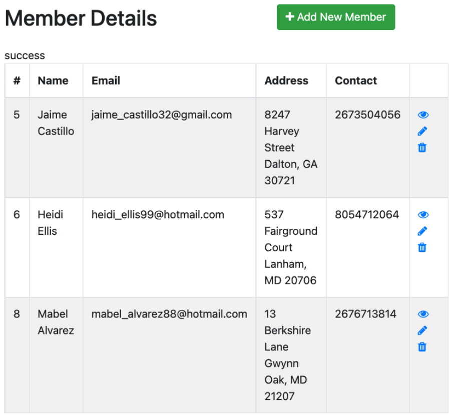
□	Edit	Copy	Delete	12	j	j@gmail.com	j	1111111111	1111 placeholder Ln

The admin may also delete members, and the member will no longer be displayed on the admin screen.



Delete Record

Are you sure you want to delete this member record?



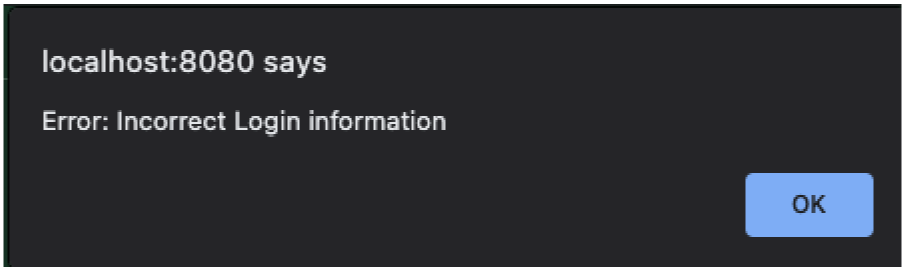
Member Details

success

#	Name	Email	Address	Contact	
5	Jaime Castillo	jaime_castillo32@gmail.com	8247 Harvey Street Dalton, GA 30721	2673504056	👁 ✎ 🗑
6	Heidi Ellis	heidi_ellis99@hotmail.com	537 Fairground Court Lanham, MD 20706	8054712064	👁 ✎ 🗑
8	Mabel Alvarez	mabel_alvarez88@hotmail.com	13 Berkshire Lane Gwynn Oak, MD 21207	2676713814	👁 ✎ 🗑

e) Data or an appropriate acknowledgment is returned to the user.

Inputting incorrect user login information results in the following message.



localhost:8080 says

Error: Incorrect Login information