# *eMarcedo* — an *eBay* clone web app

Guntakanti Sai Koushik — 180050035
Potta Nayan Akarsh — 180050074
Mekala Anmol Reddy — 180050059
Mavuri Siva Krishna Manohar — 180050057

## Requirements

This app provides most functionalities of consumer-to-consumer online marketplaces like *eBay*. Specifically, these are:

- **User Profiles & Sale Uploads** Users can make and manage accounts on this site, like adding money to balance. Seller-users can upload sales of their items on their profile for other users to see (along with some description), and they will be the auctioneer for their own products. The sellers can also upload a 'direct-buy' product to sell their product to the first buyer at the base price.

- **Bidding** When a buyer-user searches for products, they will get results filtered by location so that the auctioneer can send the products easily. They can make a bid in an automatic (limit-based) or a manual mode. They can see the best bids so far and update theirs. If the product is a direct-buy they can make the transaction immediately.

- **Delivery** When the auctioneer decides to close bids, the best is chosen. Delivery is made with status updates by the seller like in *Amazon*, and the buyer finally has to approve of the delivery for the transaction to reflect in the balance of each user, along with a receipt message to both.

- **Store Management** The employees of the application have special powers to update account balances, the status of a delivery etc. employees also have access to view sales data so that they can perform analysis on it. Administrators can (along with employee powers) add and delete new accounts, edit their details and the details of a sale.

- **Recommendations & Rating - (optional)** The buyer can rate the seller which will reflect in the average rating. This will be used to suggest new items to buy based on their history and that of other buyers. This recommendation may enhance the search results when searching for an item.
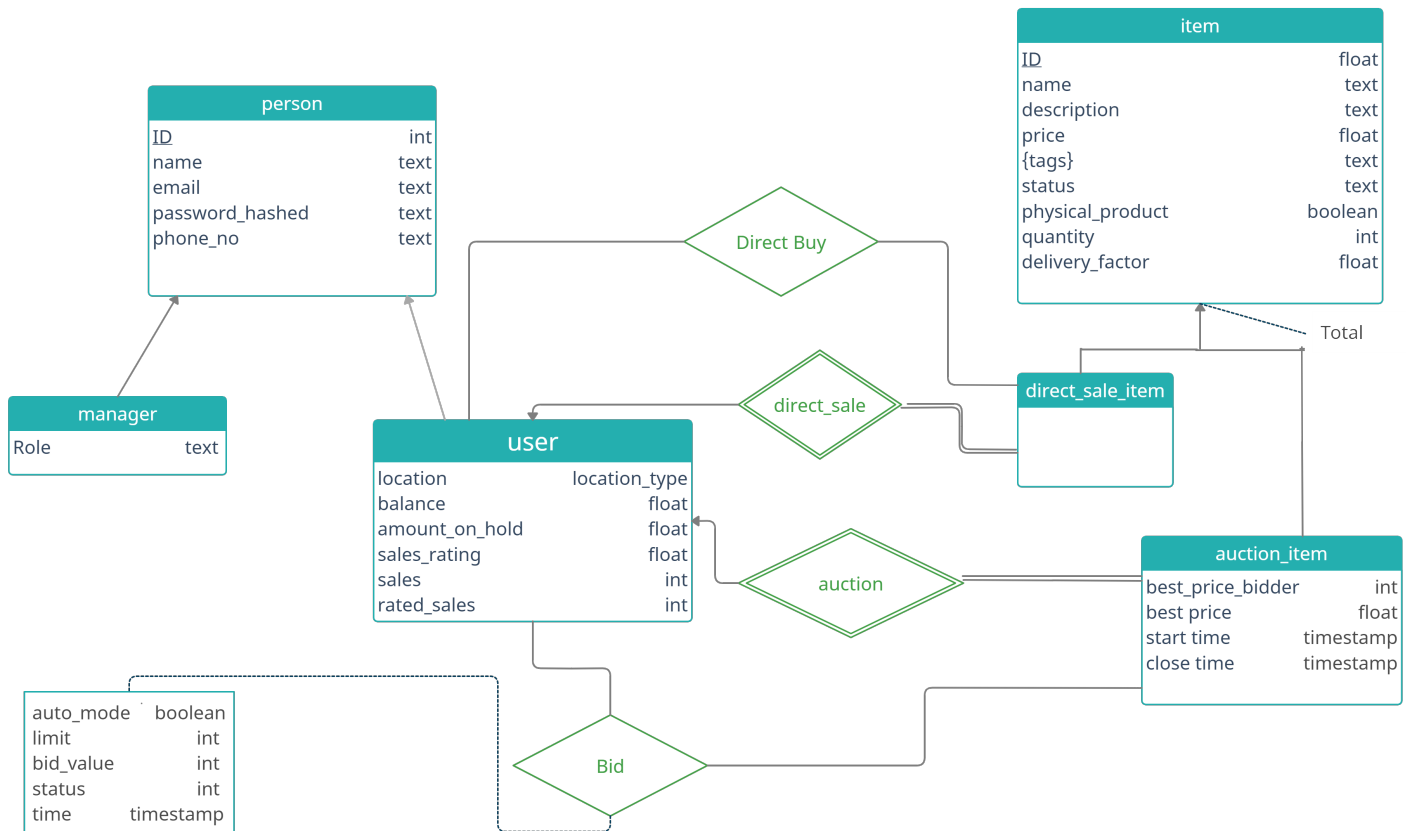
## ER Diagram



Figure 1: ER Diagram of eMarcedo

## Use Cases

- **Non-User**

  1. **Account for a user**

| Description | New user creates account |
|---|---|
| Trigger event | None |
| Primary actor | None |
| Inputs and Constraints | name, username, password, re-enter password, location, email-id, phone number; check for duplicate username |
| Preconditions | Person is not signed up and no one is logged in |
| Main success path | Creates a user account with the entered details |
| Exceptions | Verify password and re-entered password are same, if not refresh with other entered details |
| Post conditions | Person gets signed up, logged in into the new account and gets added into database (password hashed) |

2. **Account for an employee/admin**

| Description | New employee/admin creates account by giving details |
|---|---|
| Trigger event | None |
| Primary actor | None |
| Inputs and Constraints | username password, re-entered password, email-id; check for duplicate username |
| Preconditions | Person is not signed up and no one is logged in |
| Main success path | Creates a employee/admin account with the entered details. We are logged into the SQL query page which forms the entirety of pages the employees can navigate. |
| Exceptions | Verify password and re-entered password are same, if not refresh with other entered details |
| Post conditions | Person gets logged in into the new account and gets added into database as an employee/admin |

- **User**

    1. **Logging in** this one is for employee/admin too

| Description | log in to profile, or sign up with a particular role |
|---|---|
| Trigger event | None |
| Primary actor | User/employee/admin |
| Inputs and constraints | username, password; validate the inputs are matching with the database |
| Preconditions | Person is not logged in |
| Main success path | Successfully logged in and we land on the recommendations (optional feature) page for users and SQL query page for employees and admins. |
| Exceptions | Incorrect password message is displayed, Request to re-enter the correct password |
| Post conditions | No change to database |

    2. **Adding Balance**

| Description | User can add funds into his account |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | Credit Value; Non-zero amount to be credited |
| Preconditions | Account must exist and be logged in |
| Main success path | Successfully credited the entered amount, and update balance |
| Exceptions | Bad input — refresh |
| Post conditions | balance is updated in the database |

    3. **Withdraw from balance**

| Description | User can withdraw funds from his account |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | Debit Value; Amount debited must be less than (total balance - current amount involved in bidding) |
| Preconditions | Old balance is present in the database |
| Main success path | Successfully debited the entered amount |
| Exceptions | If excessive money requested — refresh with failed transaction |
| Post conditions | Balance is updated in the database as an auction item |

4. **Upload Auction**

| Description | Seller (User) can add a new item for sale |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | Product name, description, tags, base price, quantity, along with whether it is a physical product or a software, also give info about a "delivery factor" which depends on how difficult the (physical) object is to transport and which is used to determine transportation charges; name (string), quantity, price (numbers) and physical_product should not be null, and if physical, delivery factor must be a number |
| Preconditions | Seller (User) is logged in |
| Main success path | Item is up for sale and available on searching |
| Exceptions | sale is rejected if constraints like "not null", type etc. fail, and page is refreshed |
| Post conditions | Item is added to the database |

5. **Upload Direct Sale**

| Description | Seller (User) can add a new item for direct sale |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | Same as for auction but the base price here is the only price itself |
| Preconditions | Seller (User) is logged in |
| Main success path | Item is up for sale and available on searching |
| Exceptions | sale is rejected if constraints like "not null", type etc. fail, and page is refreshed |
| Post conditions | Item is added to the database as a direct buy item |

6. **View sale**

| Description | Displays all the past sales of the Seller(user) |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | View all sales button has to be pressed; None |
| Preconditions | Seller(User) logged in |
| Main success path | All the sales have to be displayed |
| Exceptions | If no sale is present then show a message telling the same |
| Post conditions | No change to database |

7. **Choose one sale**

| | |
|---|---|
| Description | Displays the details of the selected sale |
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | View view button has to be pressed; None |
| Preconditions | All the views are displayed to the user logged in |
| Main success path | Selected view's details have to be displayed |
| Exceptions | None |
| Post conditions | No change to database |

8. **Delete a sale**

| | |
|---|---|
| Description | Seller can remove the item for sale |
| Trigger event | In the bid entities related to this item, all of their statuses will become "rejected, item removed". Their on-hold balance will update removing the amount held up in bids. |
| Primary actor | User |
| Inputs and constraints | Select delete; user logged in must be owner of item, item is still under bidding |
| Preconditions | product is present in the database, belongs to current user |
| Main success path | Product removed, bidders notified |
| Exceptions | None |
| Post conditions | Product no longer available in search, removed from database, on-hold balances updated, bids updated with rejection status |

9. **Search**

| | |
|---|---|
| Description | User can search for an item in the search box by name and also add tags for categories |
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | Name of the item, tags; input not null |
| Preconditions | User is in the search page |
| Main success path | Search results appear |
| Exceptions | No relevant items found / bad input in search, shows a message stating the same |
| Post conditions | No change to database |

10. **Select item**

| | |
|---|---|
| Description | User can select a particular item from search results |
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | Click on the required item; None |
| Preconditions | search results are displayed |
| Main success path | Selected item's details have to be displayed |
| Exceptions | None |
| Post conditions | No change to database |

11. **Place a direct order**

| Description | User can place direct order for an item in search results |
|---|---|
| Trigger event | Status of item updated |
| Primary actor | User |
| Inputs and constraints | Place the order; price less than or equal to the difference of current balance and the on-hold balance |
| Preconditions | Item is up for sale and selected by the buyer (user) |
| Main success path | Order is placed successfully, on-hold balances updated |
| Exceptions | When balance constraint does not hold show insufficient funds |
| Post conditions | Status of item is updated, money deducted from balance in database |

12. **Bid**

| Description | User can bid for an item |
|---|---|
| Trigger event | If bid is better than best bid so far, their auto-bidding is triggered and their bids get updated, this will reflect in the on-hold balance of the new best-bidder |
| Primary actor | User |
| Inputs and constraints | bid value, bid mode (auto option), limit (optional - needed for auto case); bid value and limit are less than or equal to the difference of current balance and the on-hold balance |
| Preconditions | Item is up for sale and selected by the buyer (user) |
| Main success path | Bid is placed successfully, best bid and associated on-hold balances updated |
| Exceptions | When balance constraint does not hold show insufficient funds |
| Post conditions | Bid is added to the database and on-hold balance is updated based on auto-bids made, best bid may be updated in the item attributes |

13. **View bids**

| Description | Displays all the past bids of the user |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | View all bids button has to be pressed; None |
| Preconditions | User logged in |
| Main success path | All the bids have to be displayed |
| Exceptions | If no bid is present then show a message telling the same |
| Post conditions | No change to database |

14. **Choose one bid**

| Description | Displays the details of the selected bid |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | View bid button has to be pressed; None |
| Preconditions | All the bids are displayed to the user logged in |
| Main success path | Selected bid's details have to be displayed |
| Exceptions | None |
| Post conditions | No change to database |

15. **Update bid**

| Description | Updates mode of bid or value of the bid |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | New values of the same fields used for initial bids; bid value field cannot be decreased |
| Preconditions | Old details of the bid are present in the database |
| Main success path | Bid details have to be updated |
| Exceptions | Bid value cannot be decreased — refresh |
| Post conditions | New details of the bid is updated in the database |

16. **Close bidding**

| Description | Seller (User) can end the bidding of the product |
|---|---|
| Trigger event | Bids updated with final results, status of item updated |
| Primary actor | User |
| Inputs and constraints | End the bidding button should be pressed; At least 1 bid must have been made so far |
| Preconditions | All the active bids are present in the database; at least one bid had to have been made |
| Main success path | Bidding has to be ended and accepted, buyer and seller get receipts on their sale and bids, other bids are updated with rejections |
| Exceptions | If no bid is present then the item remains unsold |
| Post conditions | Status of item and bids are updated accordingly in their tables |

17. **Update delivery status**

| Description | Seller can update the item's status, buyer is able to confirm delivery |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | New value of status; It can be either Auctioned$\longrightarrow$ Shipping$\longrightarrow$ Shipped and go$\longrightarrow$ out-for-delivery |
| Preconditions | Bidding closed already |
| Main success path | Status of the item has to be updated |
| Exceptions | If already out-for-delivery or unsold, cannot be updated |
| Post conditions | New status is updated in the database |

18. **Buyer confirms delivery**

| Description | Buyer can update the item's status to Delivered |
|---|---|
| Trigger event | Balances updated of buyer and seller based on chosen bid, both get their item and bid status updated |
| Primary actor | User |
| Inputs and constraints | Buyer has to press the "Delivered" button; None |
| Preconditions | Item was in Out-for-delivery status |
| Main success path | Status of the item has to be updated to delivered, transactions completed, receipts received |
| Exceptions | If pre-condition failed, we can't update delivery status |
| Post conditions | Buyer and Seller's balance has to be updated, and Buyers on-hold balance has to be updated, status update |

19. **View previous orders**

| Description | Buyer (User) can view the list of previously bought items |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | click on view-previous-orders; None |
| Preconditions | None |
| Main success path | Previous orders have to be displayed to the buyer (User) |
| Exceptions | If no previous order is present display the same |
| Post conditions | Previous orders are shown to the Buyer (User) |

20. **Select orders**

| Description | Buyer (User) can select an item from the previous orders |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | click on an order; None |
| Preconditions | User (buyer) is in page showing all the orders. |
| Main success path | Selected order's details have to be displayed |
| Exceptions | None |
| Post conditions | None |

21. **Rate**

| Description | Buyer can give rating for a product |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | Rating; None |
| Preconditions | previously bought item has to be selected |
| Main success path | Rating is given to the selected product |
| Exceptions | None |
| Post conditions | Item's rating, seller ratings updated in table |

- **Employee** has powers to do data analysis by running queries on the items bought and update a user's balance.

| Description | Run some *SELECT* and *UPDATE balance* commands |
|---|---|
| Trigger event | None |
| Primary actor | User |
| Inputs and constraints | sql query; query validity, permission given to the employee role to do that particular query |
| Preconditions | None |
| Main success path | Query/update run and output displayed in a box |
| Exceptions | Illegal queries not performed |
| Post conditions | database updates of balance reflected in that account (there are no other side effects) |

- **Admin** has full powers to perform SQL queries and updates

| Description | Admin uses powers to run SQL queries and updates of all kinds |
|---|---|
| Trigger event | Depending on the kind of update there will be *CASCADE*s triggered which will keep the consistency intact. |
| Primary actor | User |
| Inputs and constraints | sql query; query validity |
| Preconditions | None |
| Main success path | Query/update run and output displayed in a box |
| Exceptions | Invalid queries have no effect |
| Post conditions | all the varied kinds of updates and cascaded changes reflect in the final state of the database |

**Note** Admin and employee aren't part of the ordinary usage of the application, which is purely a user-to-user interaction. They only involve themselves to directly view/update the data in certain extraordinary cases. That is why they work via SQL queries. Admin can *SELECT FROM*, *UPDATE* and *DELETE* in all relations in the entire database, while the employee can perform some data analysis by viewing the data via *SELECT* statements and doing *UPDATE*s on the user balances.

The entities we would keep in GraphDB if we implement recommendation would be the user and the items they've bought, the relationship would be the relationship found by joining each user to their item via their bid/direct-buy.

**Software**
Webpages in `html`, `NodeJS` backend, `ReactJS` frontend, `MongoDB` for geo-spatial and graph data.