

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

Guntakanti Sai Koushik — 180050035

Potta Nayan Akarsh — 180050074

Mekala Anmol Reddy — 180050059

Mavuri Siva Krishna Manohar — 180050057

## **ER Diagram (in next page):**

- This schema is in BCNF, there are no internal dependencies, except *email* being a unique attribute in *person* table.
- Since it is also a candidate key, *person* is in BCNF.
- No de-normalization has been done.
- We aren't using any triggers because we'd already manually written the updates on changing what would have been triggered in the SQL for the use cases.

describe indexing etc here



Figure 1: eMarcedo Schema

## Use Cases

- **Non-User**

1. **Account for a user**

|                   |   |
|-------------------|---|
| Inputs            | name, email-id, password, re-enter password, phone number, location, balance; check for duplicate email   |
| 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, gets added into database and goes to home page. Redirect back here for unmatched passwords or repeated emails.                              |
| SQL               | <code>INSERT INTO person(name, email, password_hashed, phone_no, location, balance, amount_on_hold) VALUES(uname, email-id, pass_hash, phno, given_loc, balance, 0.0);</code> |

- **User**

1. **Logging in credentials**

|                   |   |
|-------------------|---|
| Inputs            | email, password;  |
| Main success path | Successfully logged in and we land on the home page for users.                                    |
| Exceptions        | Incorrect password message is displayed, redirect to same page, to enter the correct password     |
| Post conditions   | No change to database, after authentication go to home page of user (“get home details” use case) |
| SQL               | <code>SELECT password_hashed FROM person WHERE email=email_id;</code>                             |

2. **Log out**

|                   |  |
|-------------------|--|
| Inputs            | email, password;   |
| Main success path | Already logged in, now logged out and we land on the sign-in page. |
| Post conditions   | User logged out  |

3. **Enter home**

|                   |   |
|-------------------|---|
| Inputs            | many use cases redirect here, clicking the logo button on top too   |
| Main success path | All current bids, sales, past buys, bids, sales and associated receipts displayed (as drop-down) and button to go into them further and get details. Buttons to upload sale, update, a search bar balance   |
| SQL               | <code>SELECT * FROM bid NATURAL JOIN auction_item NATURAL JOIN auction_item_tags WHERE person_id=curr_uid; SELECT * from auction_item NATURAL JOIN auction_item_tags WHERE seller_id=curr_uid; SELECT * from direct_sale_item NATURAL JOIN direct_sale_item_tags WHERE seller_id=curr_uid;</code> |

4. **Adding Balance**

|                   |  |
|-------------------|--|
| Inputs            | Credit Value (positive integer)  |
| Main success path | Successfully credited the entered amount, and update balance, redirect to same home page |
| Post conditions   | balance is updated in the database   |
| SQL               | <code>UPDATE person SET balance = credit + balance WHERE email = email_id;</code>        |

## 5. Upload Auction

|                   |   |
|-------------------|---|
| Inputs            | product_id, name, description, price, quantity, phys_prod, quantity, delivery_factor, tags  |
| Main success path | Item is up for sale and available on searching by other users, seller redirected to his home page, where new item is displayed  |
| Post conditions   | Item is added to the database   |
| SQL               | <code>INSERT INTO auction_item(identifier, name, description, price, seller_id, status, physical_product, quantity, delivery_factor, best_bidder, best_bid, start_time, close_time) VALUES(pid, name, descr, price, curr_uid, "open", phy_prod, qnty, d_fact, NULL, NULL, NOW(), NULL);</code> and for each <i>tag</i> given in input do: <code>INSERT INTO auction_item_tags values(identifier, tag);</code> |

## 6. Upload Direct Sale

|                   |  |
|-------------------|--|
| Inputs            | product_id, name, description, price, quantity, physical, quantity, delivery_factor, tags;   |
| Main success path | Item is up for sale and available on searching by other users, seller redirected to his home page, where new item is displayed   |
| Post conditions   | Item is added to the database  |
| SQL               | <code>INSERT INTO direct_sale_item(identifier, name, description, price, seller_id, status, physical_product, quantity, delivery_factor, buyer_id) VALUES(product_id, name, description, price, curr_user_id, "open", phy_prod, qnty, d_fact, NULL);</code> and for each <i>tag</i> given in input add the entry by: <code>INSERT INTO direct_sale_item_tags values(identifier, tag);</code> |

## 7. Click on a sale or bid

|                   |  |
|-------------------|--|
| Inputs            | Click on a sale in the in the home page or in the search results   |
| Main success path | Redirected to page showing complete description of product and option to remove sale, update sale details etc.   |
| Post conditions   | No change to database  |
| SQL               | <code>SELECT * FROM auction_item WHERE aitem_id=sale_id</code> or <code>SELECT * FROM direct_sale_item WHERE ditem_id=sale_id;</code> along with info about bid if user had bid for the item <code>SELECT * FROM bid WHERE aitem_id=sale_id and person_id=curr_uid;</code> |

## 8. Delete a sale

|                   |   |
|-------------------|---|
| Inputs            | select delete on the product description page   |
| Main success path | Product deleted from auction, triggering a bid delete, then bidders notified and their on hold balances are updated   |
| Post conditions   | product does not appear anywhere anymore, bidder gets message with details of removal on home page  |
| SQL               | WITH person_balance_item(person_id, amount_on_hold, aitem_id) AS (SELECT person_id, amount_on_hold, aitem_id FROM person NATURAL JOIN bid NATURAL JOIN auction_item WHERE aitem_id = sale_id) UPDATE person SET amount_on_hold = amount_on_hold - (select P.bid_value FROM person_balance_item P WHERE P.person_id=person_id) WHERE person_id IN (SELECT person_id FROM person_balance_item); and then DELETE FROM auction_item WHERE aitem_id=sale_id or, for direct item just delete the sale by: DELETE FROM direct_sale_item WHERE aitem_id=sale_id; (can be changed to a more efficient query by updating each bidder individually by making a separate query for each instead of doing all within a single query) |

## 9. Search

|                   |   |
|-------------------|---|
| Inputs            | item name, search tags etc entered in search bar of homepage  |
| Preconditions     | User is in the search page  |
| Main success path | redirect to search results page, both auctioned and direct sale items   |
| Post conditions   |   |
| SQL               | SELECT aitem_id, identifier, name, quantity FROM auction_item;<br>SELECT ditem_id, identifier, name, quantity FROM direct_sale_item;<br>the usage of these commands (like searching based on tag, similarity, ordering based on relevance) will depend upon how search is implemented – this is not final |

## 10. Place a direct order

|                   |   |
|-------------------|---|
| Inputs            | On the page of product details place an order   |
| Main success path | Order is placed successfully if balance available and product is not owned by oneself, on-hold balances updated |
| Post conditions   | Status of item is updated   |
| SQL               | UPDATE direct_sale_item SET status = "sold", buyer_id = curr.uid;   |

## 11. Bid

|                   |   |
|-------------------|---|
| Inputs            | bid value, bid mode, limit;   |
| Main success path | Bid is placed if balance available, best bid and associated on-hold balances updated  |
| 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   |
| SQL               | INSERT INTO bid VALUES(item_id, curr_id, TRUE, max_bid, init_bid, "running", NOW()); or, in case of no auto-bidding: INSERT INTO bid VALUES(item_id, curr_id, FALSE, NULL, init_bid, "running", NOW()); |

## 12. Update bid

|                   |  |
|-------------------|--|
| Input             | Update option available only when the user is the bidder, click on update  |
| Main success path | redirected to product details (same) page with new data displayed  |
| Post conditions   | Bid details updated  |
| SQL               | update row: UPDATE bid SET auto_mode=new_auto, bid_limit=new_limit, bid_value=new_bid WHERE aitem_id=curr.aaid AND person_id=curr.uid; |

### 13. Close bidding

|                   |   |
|-------------------|---|
| Inputs            | At least 1 bid must have been made so far, now click on end-sale  |
| Preconditions     | All the active bids are present in the database; at least one bid had to have been made   |
| Main success path | If no bid available (best_bid=NULL), redirect to same page with error message, if successful: bidding ended and best-bid accepted, buyer and seller get receipts on their sale and bids, other bids are updated with rejections   |
| Post conditions   | Status of item and bids are updated accordingly in their tables   |
| SQL               | <pre>UPDATE auction_item SET status='auctioned', close_time=NOW() WHERE aitem_id=curr_item_id; for the buyer SELECT * FROM bid WHERE aitem_id=curr_item_id; then get details of each other (failed) bid so as to update bid-status send them a message WITH rejected AS (SELECT person_id FROM bid WHERE aitem_id=curr_item_id AND person_id&lt;&gt;buyer_id) UPDATE bid SET status='rejected' WHERE person_id IN rejected; and get details by SELECT * FROM bid WHERE aitem_id=curr_item_id AND person_id&lt;&gt;buyer_id;</pre> |

### 14. Update delivery status

|                   |   |
|-------------------|---|
| Inputs            | Click on next-stage button to shift status from Auctioned→ Shipping→ Shipped and go→ out-for-delivery, (the next status is given from server based on previous status)                |
| Main success path | redirect to same sale-item page, with new status displayed  |
| Post conditions   | New status is updated in the database   |
| SQL               | <pre>UPDATE auction_item SET status=next_status WHERE aitem_id = curr_item_id; or for direct-sale UPDATE direct_sale_item SET status=next_status WHERE ditem_id = curr_item_id;</pre> |

### 15. Buyer confirms delivery

|                   |  |
|-------------------|--|
| Inputs            | buyer confirms the delivery; status should have already been at 'out-for-delivery'   |
| Main success path | Status of the item has to be updated to delivered, transactions completed, receipts received   |
| Post conditions   | Buyer and Seller's balance has to be updated, and Buyers on-hold balance has to be updated, status update  |
| SQL               | <pre>UPDATE auction_item SET status="delivered" WHERE aitem_id= curr_item_id; or for direct sale use command UPDATE direct_sale_item SET status="delivered" WHERE ditem_id = curr_item_id;, then in the next stage get the receipt from item details SELECT * FROM auction_item WHERE aitem_id=curr_item_id; or SELECT * FROM direct_sale_item WHERE ditem_id=curr_item_id;, followed by updating buyer balance: UPDATE person SET amount_on_hold = amount_on_hold-best_bid, balance = balance - best_bid WHERE person_id=curr_uid, then seller balance: UPDATE person SET balance = balance + best_bid WHERE person_id=seller_uid</pre> |

**Software** Webpages in html, NodeJS backend, ReactJS frontend, PostgreSQL (with Postgis for geo-spatial data). Our controller externalised – built using Node.js