Final Project for SW Engineering Class CSC 648-848 Section 01 Summer 2021
Team 03

Jose's Angels

Name	Email	Roles
Mitchel Baker	mbaker3@mail.sfsu.edu	Team lead
Krina Bharatbhai Patel	kpatel11@sfsu.edu	Frontend lead
Charmaine Eusebio	ceusebio1@mail.sfsu.edu	Frontend engineer
Rowena Elaine Echevarria	rechevarria@mail.sfsu.edu	Frontend engineer
Michael Schroeder	mschroeder@mail.sfsu.edu	Backend lead, Github master
Kenneth N Chuson	kchuson@mail.sfsu.edu	Backend engineer
Jamie Dominic Walker	jwalker5@mail.sfsu.edu	Backend engineer

http://dropsell.gq
August 1st, 2021

1. Table of Contents

2. Product Summary

Name of the product: Dropsell

URL to the product accessible on deployment server: http://dropsell.gq

How we plan to market and sell our product is two fold. First we will need some online advertisements to spread the name of our product. The top two methods of spreading our name brand online are pay-per-click (PPC), and paid social. PPC is one of the top methods for advertising your brand or product, but this involves competing with other big names for key search words. While this approach will work long term once we have established our name, it is not best for us at first due to the smaller purse and the fact that we are still trying to recuperate start-up costs. The second strategy of paid social is much more aligned with our goals. How paid social ads work is creating ads for social media platforms. This method allows us to target certain audiences that we feel our product applies to and since social media is one of the common pastimes of adults and teens this ensures we reach the most amount of people.

The second method to build a strong user base is word of mouth. Once we get client traffic on the website we are confident that users will want to come back again and again. As users recognize the simplicity and value of Dropsell they will quickly recommend our page to their friends and family.

What is unique about our web application drop-sell, is that we are developed from the ground up to be user friendly. Most if not all big time competitors simply added selling functionality onto their existing website. Dropsell is designed for the modern age for usability and comfort. What our web application does differently than others is that we provide tools to take the stress out of buying or selling a product. For example if you are unsure how much to charge for an item you can use our built in tool to quickly and accurately compare your product to what has sold previously or what is currently listed for sale. This takes the stress of guessing a price and underselling the product.

Itemized list of functions

1. Users for this application will be able to search for products.
2. A person that wishes to sell an item can choose to list that item directly to the market place or post it as an auction.
3. A user will be able to purchase an item by auction or buy it now style options, whichever the seller enables.
4. A user will be able to sort and filter search results to best suit their needs or interests.
5. A user will be able to see details related to their purchases in their account information.
6. If a user opted for delivery consignment upon purchase of an item, then they will be able to see tracking information in their account.
7. Instead of face to face deliver options, users can elect to have items delivered via consignment options(AKA shipping via ups for example)
8. A user shall be able to use website tools to price match items they wish to sell or purchase.
9. Any user must agree to the website's terms and conditions.
10. A user can message sellers with questions related to their products.
11. Users can add items to their shopping carts.
12. A user's shopping cart will update as the user adds or removes items from the cart.
13. A user shall be able to return back to shopping to further fill up their cart if they are midway through the check out process.
14. A user can choose to cancel their order or modify it appropriately.
15. A user can have details related to the purchase sent to their accounts for review at a later date.
16. When a user is checking out they can choose to add any additional information they want
17. When a user checks out they can choose different forms of payments.
18. A user shall receive a receipt/invoice after purchasing their item that will include all information required(ie, purchase price, name of seller, contact info, etc).
19. Users shall agree to terms and conditions prior to be granted access to

20. Each user shall agree to a seller's fee that will be deducted once that item sells.
21. A user will be able list products for sale and will list any pertinent information.
22. A user shall be able to edit a post that they made if they deem it appropriate.
23. If a user wishes to sell more of a certain item they will be able to adjust the quantity on their post.
24. A user shall be able to delete or remove an item if they no longer wish to sell it.
25. A user will be able to check all items they have for sale in their profile/account page.
26. A user will have different options for selling products(IE a user can choose the starting bid for an item, a user can choose if that item should last for 1-3 days, etc).
27. A user can set an item’s duration on the website for a minimum of 1 hour, to a max of 30 days.
28. A user will be able to see pertinent information related to an item they are interested in purchasing(IE a user can see the current bid on an item, a user can see time remaining on an item, etc.).
29. A user shall be able to bid on an item with a single click.
30. A user shall see an accurate count down of time remaining on an item's listing time.
31. Users shall be able to contact other users.

3. Milestone documents M1-M4

M1

SW Engineering CSC648/848 Summer 2021

Jose's Angels (Dropsell .gq)

Team 03

Name	Email	Roles
Mitchel Baker	mbaker3@mail.sfsu.edu	Team lead
Krina Bharatbhai Patel	kpatel11@sfsu.edu	Frontend lead
Charmaine Eusebio	ceusebio1@mail.sfsu.edu	Frontend engineer
Rowena Elaine Echevarria	rechevarria@mail.sfsu.edu	Frontend engineer
Michael Schroeder	mschroeder@mail.sfsu.edu	Backend lead, Github master
Kenneth N Chuson	kchuson@mail.sfsu.edu	Backend engineer
Jamie Dominic Walker	jwalker5@mail.sfsu.edu	Backend engineer

Milestone 1

June 22, 2021

History Table

Version	**Date**	**Notes**
M1V2	07/02/2021	
M1V1	06/22/2021	

1. Executive summary

Last year the world was shocked as we transitioned from the idea of a global pandemic being an outdated science fiction trope, to full blown reality. The various nations of the world urged social distancing and staying at home till the crisis could pass. The challenge is that as great as it is to stay home from work or school all day long, eventually you get bored and want to do some shopping.

Gone are the days of waking up early on the weekend and driving around town and looking for the best bargains at the local yard sales. These days it is all about finding what you want when you want it. The issue is that the market is flooded with half-baked ideas that have evolved to make ends meet. Facebook was initially intended as a way for college students to keep in touch. eBay was designed as an auction site 25 years ago, and Amazon started as a bookstore. There is not a major digital marketplace that was designed for the current state of the world.

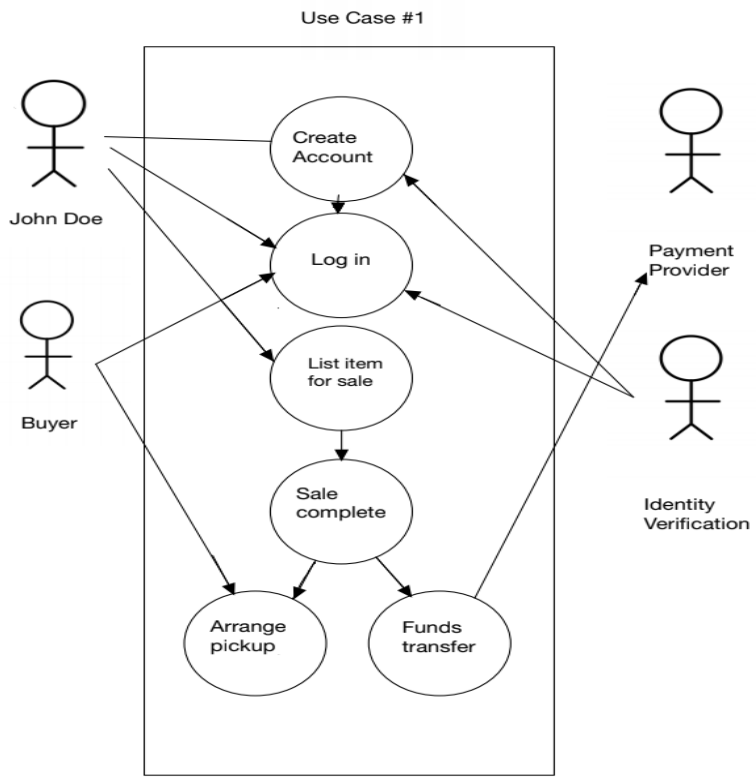
Fortunately for you (and introverts all over the world), we at Dropsell are developing the next generation digital marketplace that is designed for the world you are living in now. It is this innovative mindset that sets our team apart from the competition. We focus on maximizing the user experience with industry leading best practices so customers can shop safely and securely. We will offer a simple clean digital storefront that makes the focus all about buying and selling exactly what you want or need. We at Dropsell understand that buying or selling for the first time from another person can create unneeded anxieties and we want to ensure sellers have all the information they need to make safe and informed sales. For example, maybe you have a brand-new PlayStation 5 you want to sell but are not sure how much it goes for. No need to price check on Amazon or Walmart, we include a price checker for you when listing an item to cut down on the amount of research you must do.

The beautiful part of this project is that we are providing a platform for buyers and sellers to populate with their own content. Once initial development is complete, 90% of our capital will be towards maintaining the application with a smaller maintenance team. What this means for our investors is that with competitive service fees for transactions, coupled with low day to day costs, we can expect a steady return of investment. While this plan will not make everyone rich overnight, it will provide a steady income for years to come.

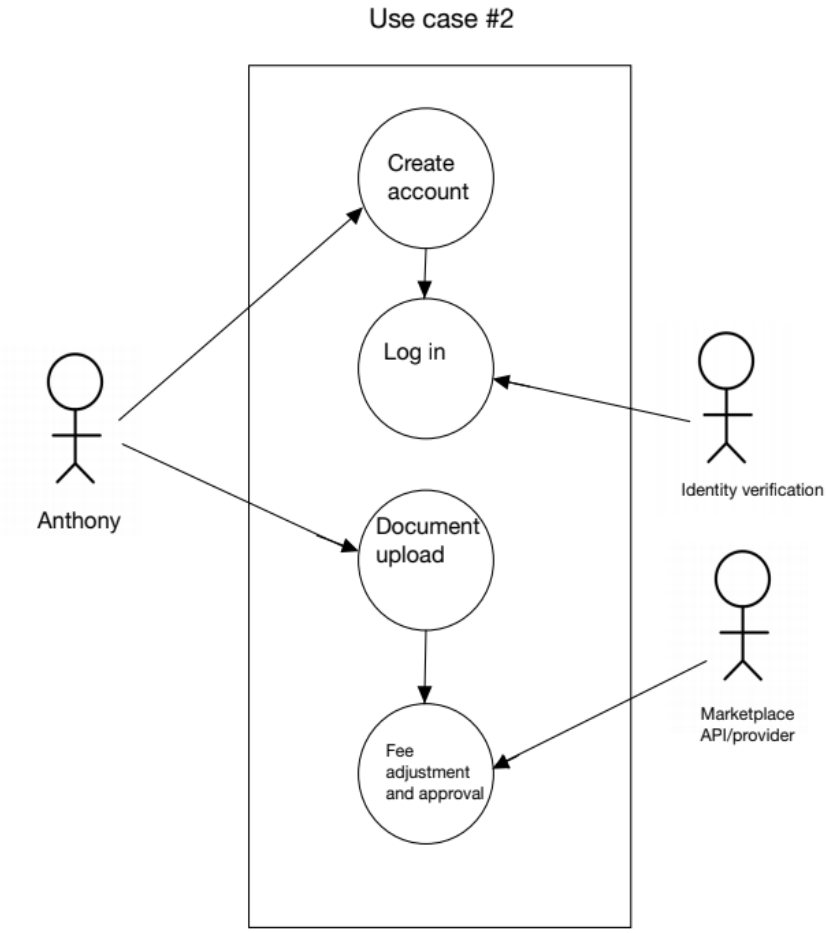
Here at Dropsell , we have an experienced group of buyers and sellers all working hard to bring you the best possible experience. So, if you want to learn how to upgrade your yard sale to a streamlined marketplace, let us show you how.

2. Main use cases

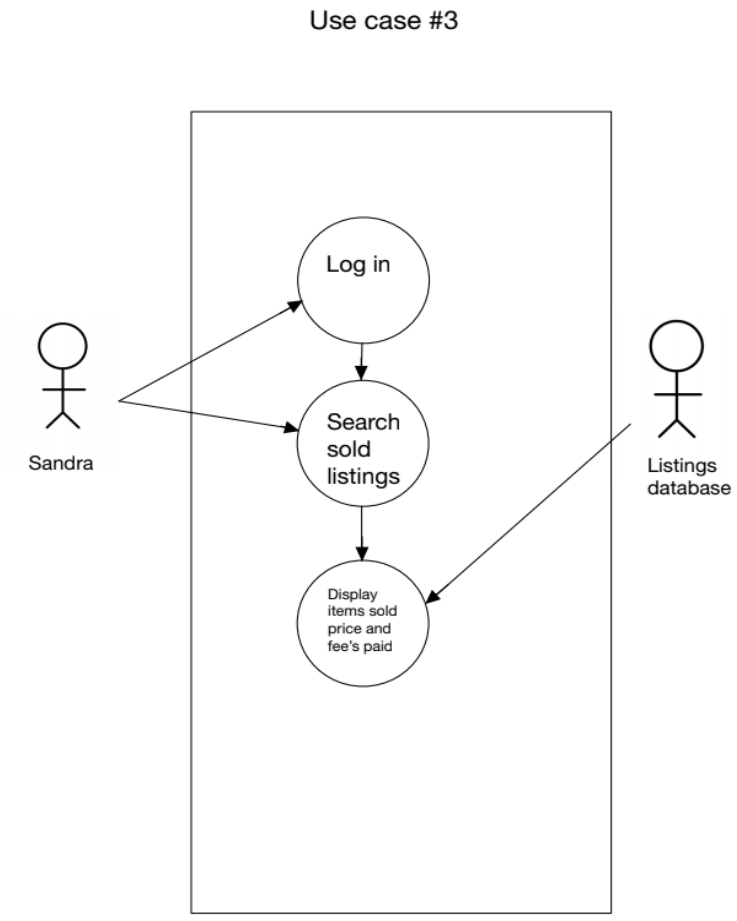
Title	Easy Onboarding Experience
Actor	John
Description	John is moving into San Francisco. He is in the process of selling some of his old belongings which range from computers, gaming devices, to couches and other kitchen appliances. John hopes to find buyers who offer a reasonable price. As a result, he will be better positioned to finance his upcoming transition to a new city. John has been active with eBay and Facebook marketplace for a while, but he has not had any success with potential buyers. John is moving in the next 3 days. It is no big deal if he cannot sell the small items such as the computers, gaming devices, kitchen appliances since he can always include them in his moving truck to sell in San Francisco. On the other hand, John must sell the couch because his other options are taking it to the dump or donating to goodwill which will not turn him a profit. John is looking to find a buyer so he can make a quick buck, and since he is leaving in 3 days, as fast as possible.



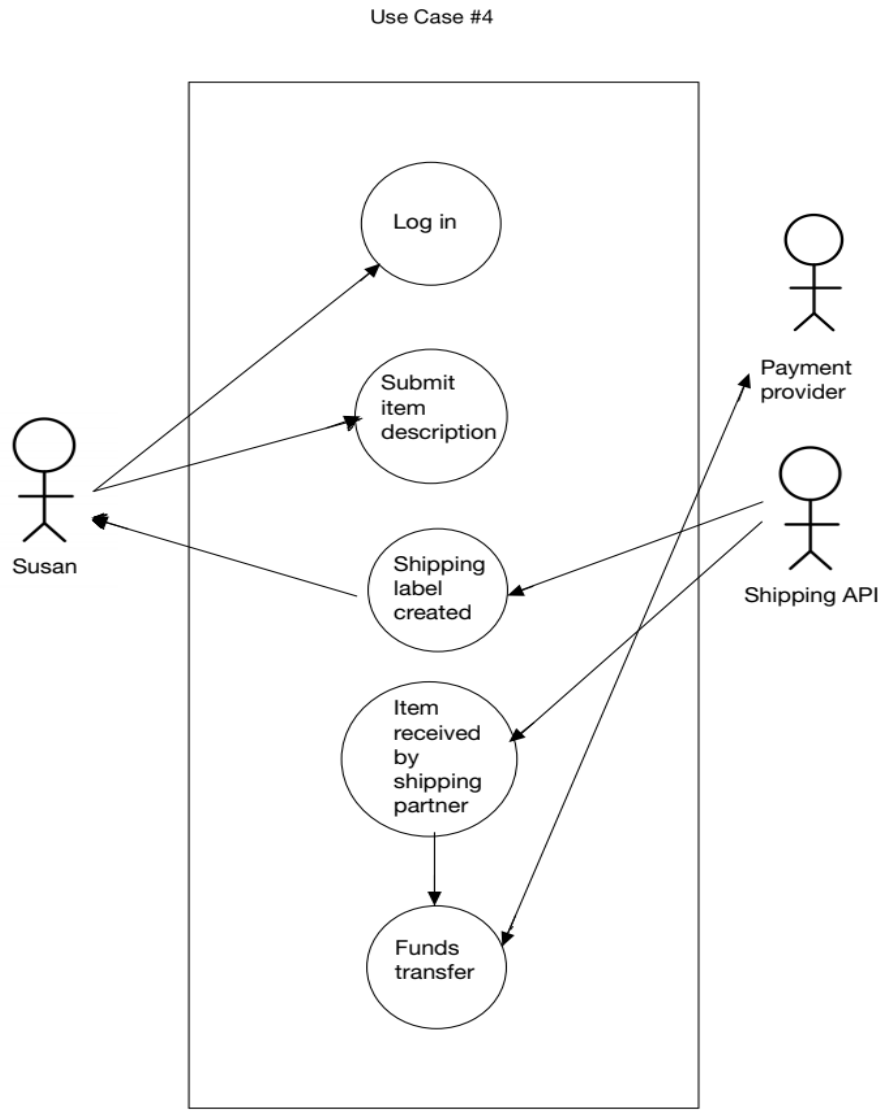
Title	Storefront Features with less fees
Actor	Anthony
Description	Anthony wishes to sell items for the long term. He is looking to provide reasonable deals through his brick-and-mortar location, but the storefront subscription prices are too much for him to start out. Anthony wants to prove he is a small business owner so he can access storefront features and lower fees on our website, instead of a competitor.



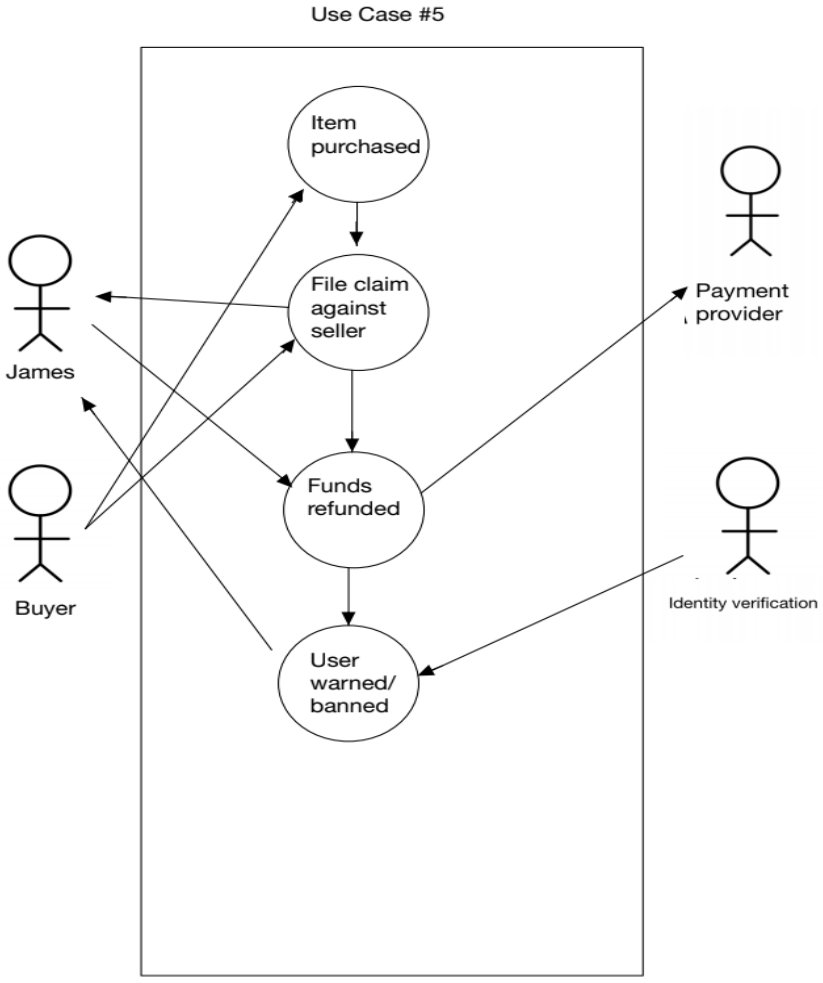
Title	Price Matching Feature
Actor	Sandra
Description	Sandra, a buyer, has the old iPhone 10 model, but they are busy with work and do not have time to go to a brick-and-mortar store. As a result, Sandra needs an easy-to-use website which allows her to get the new iPhone 12 and avoid scalper prices. She checks the price of a new iPhone 12 on Facebook Marketplace, Amazon, and eBay, but all these prices are out of their price range. Sandra can check our website to compare recently sold listings and fees for a smart buying decision.



Title	Shipping available with extra benefits
Actor	Susan
Description	Susan is doing some spring cleaning in their home and is too busy with work to be able to spend time devoted to creating individual listings for each of these items. This allows the consignment service to provide the seller with the option of passing over the creation of listings and prices to us.



Title	Flagging system to avoid fraud deals/scams
Actor	James
Description	James is in the business of fabricating fake PS4 controllers, which he then markets to people as genuine. However, his fabricated replicas are only plastic shells which means they have no functionality whatsoever. James creates an account on our site and begins creating listings for these fake PS4 controllers. Solution: If buyers begin to leave bad reviews for these fake PS4 controllers, they can report James (seller) to us, and we can flag the item as a scam so the listing can be removed.



3. List of main data items and entities

API Data

1. Google maps for locations and places. Users should be able to interact with a 2D map where they can see the locations of sellers and the products they have posted onto Dropsell .
    a. If items are listed for pickup, then users should be able to filter google maps so they only see pickup items
2. Items from e-commerce websites (Amazon, eBay, etc.) for item comparisons and price matching. We will have to use an external API to provide users with the best prices for items listed on Dropsell .
    a. Price matching can be used for buyers to notify them about cheaper deals or better prices as opposed to our competition.
    b. Price matching can also be used for sellers to notify them what a good price would be when listing a new item. If a seller lists an item with too high of a price, they may never get a buyer. Conversely, if a seller lists an item with too low of a price, they may get swindled out of proper reimbursement.
3. UPS developer kit API
    a. If buyers request for their item to be delivered, sellers can easily create a description of their item and send it off to a delivery provider so that a shipping label can be easily created.
4. Our own Node API server which will return requested data in JSON format to the react client.
    a. The Node API server will load user data, post data, messaging data, customer reviews and ratings, seller reviews, and for storing shopping cart data securely.

Unregistered User

1. An unregistered user should have the freedom of exploring Dropsell , in terms of the products displayed in the marketplace and public facing pages open to all users.
    a. However, unregistered users cannot create comments.
    b. Unregistered users cannot rate products or sellers.
    c. Unregistered users cannot participate in auctions.

Registered User

1. User login (active session)

   a. The user's active session will be used to record the date and time when a user has logged in and to record the length of their current session.
   b. When the user logs in for the first time, their session data should be updated in the database. If their session has expired, then the user should be required to log in again. If their session is still active when they refresh a page as an example, then the user should stay logged in.
2. User logout (inactive session)
   a. If the user decides to logout from Dropsell , then their session should be destroyed and declared inactive.
3. User first name, User last name
   a. The user's first and last name will be used when they decide to either purchase a product from the marketplace, or when they decide to create shipping labels using our consignment operations. We will prompt the user the first time they are in the process of making a payment for this information.
4. username
   a. A user's username will be used to login the user. It will also be displayed in messages between other users depending on whether they are a buyer/seller.
5. email
   a. A user's email will be provided upon account creation. Their email will be used to reset their password or other account information if the user happens to forget this.
   b. A user's email should be unique, and it should also be in proper email format with @
6. phone
   a. A user's phone number is important for sending them notifications from Dropsell . If a buyer bids a user's auction, then they should be notified about these actions.
7. age
   a. The age of the user is important for determining whether they are old enough for using Dropsell . Users should be at least 13 if they would like to create an account, and 18 if they'd like to sell products.
8. user password
   a. A user's password will be used to log in. If a user forgets their password, then this data will have to be reset in the database.
9. user messages

   a. User messages should be an array of objects, with the objects representing the messages themselves.

10. user street, city, state, zip
    a. If the user chooses delivery when checking out for the first time, then they should be prompted to enter their street, city, state, and zip code. If the user has checked out before, then they should have the option of choosing an existing delivery address.
11. user voice/video calls
    a) Users should be able to communicate between other users prior to their meetup times.
12. user meetups
    a.) Users should have meetup plans between other users.
13. user payment information
    a. By using the Stripe.js API, users should be able to securely store their CC number, expiration date, 3-digit code, and zip of their card. They

User Messages

1. Should be an array of objects, with each object containing the ID of the message poster, the ID of the message itself, and the time that the message was sent. The user's messages array will be used to keep track of all the message history the user has had.

User Meetups

1. Users should have clear indications of who they are meeting, the location they are meeting, and the time they are meeting with a potential seller.

Buyer

1. By default, users should be considered buyers. That way, they can immediately begin browsing the marketplace to purchase products.
2. customer refund/return products
   a. Buyers can be refunded for the products they have purchased as long as the money back guarantee is still valid.
   b. Buyers should be able to return products back to the seller if they are not satisfied.

Buyer Shopping Cart

1. The buyer's shopping cart is important because this is the only way to provide users with a secure and real-time way of tracking the products they would like to purchase.
    a. Buyers should first and foremost be able to add multiple items into their shopping cart.
    b. Buyers should see a subtotal of their products.
    c. Buyers should see the quantities of each product.

Seller

1. To become a seller, user's must provide their driver's license or some other form of identification to become verified as a seller.
2. When a user decides to become a seller, they should be prompted for their location.
    a. Sellers must have a location to let buyers know where they should meetup.

Seller Ratings

1. All sellers will have a 5-star rating system associated with their account. This way, good sellers will be rewarded with their hard work. The higher a seller's rating is, the more they will stand out to buyers.

Seller Products

1. After a seller inputs all the required information, then their new product will be created.
    a. Each product should be labeled as an auction item or marketplace item.
    b. Every product should have a clearly defined name.
    c. Every product should have a clearly defined price.
    d. The quantity of the products being added should also be defined.
    e. The product should have images associated with it to provide buyers with multiple angles.
    f. If necessary, products should have a selection type, as in different color options.
    g. The product's brand name should be added.
    h. The year the product was made should be clearly defined.
    i. The product should be recorded as new or used.
    j. Each product should record the number of interactions it has had.
    k. Each product should record the number of purchases it has had.

Product Discussions/Comments

1. Buyers should be able to have discussions about the products by creating comments and chatting with other buyers. Previous buyers of the product can let new buyers know about product experience, how the delivery process went, or just share general comments about how they have enjoyed using the product since receiving it.

Product Ratings

1. Buyers can rate products they have purchased in the past using a 5-star rating system.
    a. Buyers should have a complete breakdown of all their past ratings.

Product Refunds/Returns

1. If a buyer is not satisfied with their purchase, they have the option of receiving a refund for the item if they submit a request within the first 30 days. If a buyer's purchase has been lightly used or not opened, then buyers also have the option of returning the product they have purchased.

Auction Products

1. If a seller lists a product as an auction item, then these products should be linked accordingly in the database to an auction products table. By creating an auction products table, we can group products into either auction or marketplace items.

Top-Purchased Products

1. On the homepage, buyers should be able to interact with the topmost-purchased products to see what others are buying. These products are the ones bought most frequently by other buyers, so it makes sense to display these to the user because they may also find utility in the product.
    a. There should be a limitation of around 10 top-purchased products.

Daily Deal Products

1. On the homepage, buyers should be able to interact with products that are posted with reduced, short-term prices. Maybe there is a flash 24h deal on Dropsell which could provide an insane discounted offer for a product a buyer has been looking for. By displaying the best offer products on the homepage, buyers can quickly see these and make a decision.

Similar Products

1. If a buyer has spent a considerable amount of time exploring the products available on Dropsell , then they should be shown products which are like the ones they have viewed previously. Maybe a buyer was not able to find the most ideal price for the product they were looking for. If they check the similar products section, they could find a better price for the product but with a different brand or they could find a seller with a better offer.

For Sale Products

1. If a seller decides to reduce the pricing on their products, then these products should be shown here to the buyer. Buyers may want to purchase a cheap product which is why the for-sale products should be displayed to them.

Shipping Products

1. We will be providing our users with consignment operations to make the shipment of products easier. When a product is ready to be shipped, users should be provided with consignment data related to any barcodes or labels to be created, product's shipment information such as addresses and phone numbers, the departing and arriving locations, the time the shipment was sent out, in addition to the estimated arrival time.

4. Initial list of functional requirements

Marketplace

1. Users should be able to interact with a search bar for looking up new listings.
2. Users should see an advanced search bar for more detailed searches.
    a. Users should be able to choose from a broad range of details from size, price range, color, etc.
3. Create an auction, buy it now, or best offer options for listing.
    a. More options for sellers to earn money.
4. Share listing feature to share products with friends, through email, social media, or a link.
5. Buyers should be able to interact with sellers using the google maps API.
    a. Buyers should have a clear indication of their general location in correlation to the sellers. They should also be provided with travel times.
6. If a seller has different styles of their product, i.e. colors or fabrics, then users should be able to choose their preferred style.
7. Buyers/sellers should be able to schedule/edit meetup times if they agree with each other.
8. Users should be offered a base price for automatic reject or accept.
9. Users should be shown ads of similar items on the product page.
    a. To attract more customers to buy products.
10. Users should be shown a shipping fees table.
    a. Breakdown of the shipping fees for time efficiency
11. Order status: confirmed, processing, shipped, returned. This will allow the user to track the status of their order/s.
    a. The user should be able to get immediate updates on the status of their order. They should also be able to check their order history, and then check on specific orders to see its processing, shipping, or return details.
12. Users should be able to track the delivery of their package.
    a. This will allow users to confirm that they have received their order.
13. Display “out of stock” for empty selling items.
    a. This will let users know that the item is no longer available; they will then have the option of being contacted by email when the item comes back in stock.
14. If a product is purchased multiple times by the same user and it happens to become sold out, then the user should be notified and refunded if necessary.

15. Flagging system for reporting suspicious items. Users can either flag the item as inappropriate, a scam, or they can continue the process further by submitting a ticket to us for further review.
16. Wedding registry for those who wish to create a Wishlist for their special day. This will allow users to add products from the app into a list that they want to share with others.
17. Users should be able to go back to items they have recently viewed.
18. Must show the maximum time length shipping delivery item.
19. Must show the minimum time length shipping delivery item.
20. User monitoring so that we can display to the user the posts they recently interacted with. I.e., keep track of the IDs of the posts they click on, and before redirecting to another portion of the app, send a post request to the server which will add these items to the database for interaction later.
21. Show people who have the same purchase as you.
    a. When a customer purchases an item, the feature will pop up the list of customers that have the same purchase and a customer will be able to message or call one of the customers about relating to the similar purchase item.
22. If the buyer's and seller's home location are closer around 1 or 2 miles, then the algorithm will hide the time length of the shipping delivery item.
23. International marketplace- buyers can buy products wherever they are in the world.
24. International marketplace- sellers can sell products wherever they are in the world.
25. International marketplace- currency converter

Auction

26. Filtering for item searching, min/max price, location filtering (within 50 miles), type of shipping (pickup, delivery. This will help buyers and sellers to choose the type of delivery options.
27. Sellers should be able to have full control over the settings of their auction.
    a. They should be able to set the average starting bid, the total duration of the auction (1, 3, 5, 7 days), and the time when the auction begins.
28. When an auction is posted for bidding, users should be able to see the current price it is set at, the amount of time remaining in the auction, and any bids from others.
    a. To make auctions real time, we will have to set up some sort of WebSocket lobby where everyone partaking in the auction has access to the auction's stats.
29. Buyers can bid on the products as many times as possible.

30. Buyer can bid on products with one click.
31. Buyers can watch products for auction.
32. Buyers can see how many other buyers are bidding on the same product.
33. Buyers can see how many bids the product has.
34. Buyers/sellers can see the live countdown of the auction.
35. Buyers will get notified when the auction is 5 minutes from being over.
36. Product listing will have time until the auction is over.
37. Sellers can set the auction to go as long as 30 days, or as short as 1 hour.
38. Buyers cannot retract their bid.
39. If a user wins an auction, they should be notified immediately, either by text or by email. (email seems more doable at this point, texts with Twilio are added cost)
40. After the user wins an auction, they should be notified and then provided the option of choosing from multiple payment options.
    a. The user can pay with credit card, Paypal, or ApplePay/GooglePay (if time)

Messaging

41. Contact the seller directly on the website.
42. Contact the seller via email.

Buyers

43. Purchase agreement to be signed by every user intending to make a purchase on the app. The buyer will agree to conduct business according to laid out rules.
44. Buyers should be able to seamlessly send an inquiry of interest or any questions to the seller of their choice.
45. All the user's selected items to be purchased will be added to their shopping cart.
46. Buyers should be able to remove an item from their shopping cart, and the total list of items should be updated. Buyers should also be able to return to the main shopping menu to look for other items if they are not done.
47. Cancel/modify order option. This will allow buyers to cancel their order or make changes in case they would like to add more items to their cart.
48. Buyers will receive a detailed receipt of their purchase with tax and total after checking out through email.

49. Buyers will receive a confirmation email of their purchase with shipping information.
50. Buyers should be able to click on the product listing images to zoom in and see more details of the product.
51. Buyers should be able to add their receipt information after checking out with their cart for the first time.
    a. If they are new users, have them update their receipt info. If they are returning users, then they should be able to choose their existing receipt info or add new credentials.
52. Buyer payment option, (if user is buying for the first time, then have them update their payment option, if they have opted to remember this information, then payment will already be there)
53. The buyer should be notified by final invoice of their purchase's pickup time/delivery time, expected time of arrival, name of seller, location to meet them at, their contact info, and the final total to be paid.
54. Purchase/order history for buyers to keep track of their past purchases.
55. Buyers should be able to rate/star products to keep track of them.
56. Buyers should be able to choose delivery options.
57. Buyers should be able to interact with the seller's review page.
58. When a user clicks on an item they would like to see more of, display not only the photo slider but also the ability of messaging/sending an inquiry to the seller. (UI MODIFY)
59. Product reviews on product listing page. The buyer will be able to submit a review for purchased items.
60. Favorite's list - the buyer will be able to add what they are interested into an ongoing list.
61. Wishlist. This will allow the buyer to save the items they would like to buy in the future.
62. Discount codes for frequent buyers. The codes will have expiration dates and will attract business by providing special discounts for potential buyers.
63. Coupon/promo code box for the buyer to enter the code and have the discount applied to their shopping cart.
64. Watch item feature/list. If a product is in an auction, the buyer will be able to keep track of the most recent updates to the item.
65. Buyers can have an award item.
    a. If a buyer bought some lucky items, then the buyer will have one of the free award items.
66. Birthday promo code / random award item.

   a. If a buyer's birthday is today, then the buyer will choose between a Promo code or an award item from a random item generated algorithm.

67. Generate a referral URL which existing users can send to their friends.
68. Users should be able to subscribe to a specific seller so that they can be notified if the seller adds new items, adds updates, or makes any changes to the current price of an item.
69. When a buyer purchased an item, it will show some top-rated items that are related to that buyer's purchase item.
70. Email order confirmation to both buyer and seller. The email will contain information of the product sold, time it was sold, and who bought it.
71. Buyers should be notified about the items they viewed previously, and they should be asked if they are still interested in these items.

Sellers

72. Sellers should be able to create a new listing with a name, description, image, etc.
    a. Each listing also needs their own respective ID
73. Sellers should be able to edit the description, title, and price of their items. This will allow the seller to advertise their products better.
74. Seller contract to be signed by all users who wish to sell. This will provide sellers with an outline of how to sell on the app and rules they must follow.
    a. Seller listing fees for each product that is sold. This will include a percentage of each sale being taken as profit for the app.
    b. Seller background checks for all users intending to sell on the app. This will help to fight against scammers potentially.
75. Sellers should be able to adjust the quantity of an item that is available for sale.
76. Sellers should be able to delete an item. This will allow sellers to delete any items they think that will not be able to be sold.
    a. Ability to have multiple select delete items.
77. Under the seller section of the user's profile information, users should be able to keep track of all the items they have posted.
78. Storefront profile page for sellers. This will showcase some products that the seller has up for sale, along with some details about who the seller is or what their shop is about.

79. Seller tool - relist item. This will provide sellers with the option to easily relist their product being up for sale.
80. Seller tool - send offer to specific buyer/buyers. This will allow sellers to gain potential business from buyers who are interested in their products.
81. Each seller's product should have its own review page.
    a. To provide buyers with insight into the functionality or state of the products being bought on our site.
82. Each seller should have their own review page/star rating.
    a. Proving seller's credibility
83. If one of the items has very good ratings, then the feature can hide ratings.
84. If the seller's item(s) is/are illegal, then it is the seller's responsibility to display "illegal item", otherwise the seller can have a chance to get into trouble.
85. While displaying "illegal item", then the seller must explain the illegal item.
86. Sellers will get email confirmation that one of their products has sold.
87. Seller's product listing will reflect when an item is almost out of stock.
    a. "Only a few left in stock" or "Last one available"
88. Striking system for users for wrongdoing. This will allow the app to prevent users from continuing misconduct or violations of the terms of service. Perhaps 3 strikes on a user account will lead to a ban.
    a. User ban/blacklist

Website Features

89. Consignment operations, either for having items delivered or for meetup with the seller, or from a designated pickup location.
    a. Freedom to choose buyer's delivery options.
90. Sourcing form (daily item forecast), profitability test, how much it costs to ship the product as opposed to being shipped on other services.
91. Price matching, for items either already posted on our app or compared to other items on other websites (amazon, craigslist, eBay)
92. Users should be able to take advantage of an algorithm which keeps track of how many interactions their listings have.

    a. There will be functions attached to each listing which will send/receive data from the node API to create these real-time updates.

93. Item hashtags to help with categorization. Sellers will be able to use hashtags to categorize their products. All users will be able to search products using hashtags.
    a. To see the best deals.
94. For each item a seller posts, the user should be able to track how many interactions it's had, if there are any people who have reached out for the item, in addition to how well rated the actual item is.
95. Buyers should be shown items which are similar in price or category to the item being viewed. A comparison between the item specs could be displayed as well.
    a. As an example, let us say that you are looking to buy an USB-C adaptor for connecting a monitor to your computer. When clicking on one USB-C item, users should have the option to compare the price, item features/details, and specs in depth to USB-C adaptors of similar brands so that they can make the most accurate decision for what they are looking for.
96. Daily deals which will highlight the sales going on and attract potential buyers to listings.
97. Sellers should be able to see the statistics related to how many buyers have purchased their products.
98. The seller should be able to keep track of the median price ranges of items similar to the one they are listing
    a. For example, Dropsell has a few tables listed in its marketplace which have been set to an average price of $500. If a seller lists a table and he/she sets the price to $100, then Dropsell would notify the seller that they may be able to get a better profit by increasing their price.
99. Users should be able to zoom in or zoom out of an item with a magnifying glass feature
100. Tax calculated automatically based on buyer origin. This tax will be reflected at the checkout process page.

5. List of non-functional requirements

Networking

1. There should be WebSocket functionality for displaying the most recently created posts, the hottest daily pics, and other displays of item data.
    a. WebSocket functionality should also be used for displaying user's notifications, messages, and for auctions.

Security

2. SSL certificate, which enables websites to move from HTTP to HTTPS, for better website security.
3. Warning/caution list page for what sellers/buyers should look out for. A standard list providing details and warnings for potential scammers or bots.
4. Submit a ticket. After flagging an item since it seems sketchy, or an item might be fake, the user could continue the process by filing a ticket and sending a message to us to follow through further with reporting suspicious behavior.
5. Limited login attempts for security purposes. For example, if the user enters the wrong password 3 times, they will be locked out of trying to log in to their account for 30 minutes.
6. Captcha to fight against bots. Captcha provides an extra level of security for users.
7. Email verification for both seller and buyer. Verification ensures that there is a valid user behind each seller and buyer account.
8. Two factor authentications for added security measures. The user will be required to authenticate their account on two different devices to help prevent a breach in security.
9. Use of sessions on the backend to securely keep track of the user's shopping cart information once they redirect away from the main shopping site.
10. Users should be able to view and update their personal information settings securely.
    a. Users can update their account information, change their full name, update their email, change their password. Update their payment information or see their complete payment history in a secure fashion.

11. Password recovery for all users when they need it. The recovery process will involve having an email sent to the user to the email address associated with their account, where they would have the opportunity to change their password.
12. Username recovery for all users when they need it. Like the password recovery process, where the user will get an email to recover their username.
13. Load balancers or distributed microservices on the backend so that we do not have just only servers doing all the backend work? We do not want the backend to go down and have the whole server unavailable to all users.
14. Keeping track of unaccepted/accept usernames and writing reviews.
    a. They must be appropriate and cannot contain duplicate usernames.
15. If a user does not do anything while logging in for like 15 minutes or more. Then it automatically logs out for the user.

Server-side validation

16. Correct login information entered. Username and password matching correctly.
17. Valid post information when a seller creates a new post.
18. The user's shopping cart should be updated with the total amount of the user's cart and their selected items FROM THE SERVER. User shopping cart data cannot be handled by the client, the client should only be used to display the data sent from the server.

Client-side validation

19. Username should be at least 8 characters in length. The username shall have a minimum length for security reasons.
20. User password should contain uppercase/lowercase and numbers with a minimum length of 8 characters. This password requirement ensures an extra layer of security.
21. When a user inputs their email into the registration form, a valid email should have @ along with a valid ending. This requirement checks if the email is in the right format.
22. New post title should not exceed 80 characters in length.
23. New post description should not exceed 500 characters in length.

Password encryption

24. A minimum of 10 salt should be used when hashing a user's password.
25. Perform password hashing when a user creates a new account. This will guarantee us that passwords saved in the database are secure.
26. Bcrypt library should be used for hashing a user's newly created password.

Authentication

27. A user's session should be serialized after logging in, and deserialized when a user leaves the site.
28. Stripe.js or PayPal payment SDK should be used for creating and validating payments from users.

Database

29. Seller analytics page (demographics of customers, what is sold more frequently, etc.) Analytics will provide the seller with information to help them better their product listings.
30. Passport library should be used to authenticate an existing user's requests.
31. Encapsulate any SQL statements so that SQL injections are not possible.
32. The table for seller posts should have the following constraints for its ID column: integer type, auto increment, and public key.
33. Each post should have a foreign key tied back to the user's id who created the post.
34. Each post should have columns for the posts' title, description, price, photo, the time when the post was created, if the post is still active on the app or whether it has expired.
35. Post comments should be represented in a comments table. Each comment's id should be of integer type, auto incremented, and the public key.
36. Each comment should also have a foreign key tied back to the creator of the comment, and a foreign key tied back to the post the comment was for. Each comment should also have the time that it was created.
37. Session data for each user should be stored in the database. If their session is still valid, allow them to log in directly into the app.

Usability

38. Redux should be used to manage the state for the react client. Redux will be essential for keeping track of user's shopping cart information.
39. Use of Redux mapStateToProps() for updating user's shopping cart information, loading user account details,
    a. The mapStateToProps() function will be used to load state data, such as a user's shopping cart items as properties passed into React components so that the data can be displayed to the user.
40. Multer library should be used for uploading images to the server/database.
41. Browser support for IE, Chrome, Firefox, Brave, etc. Cross-browser support for all users to be able to access.
42. By default, users who are not logged in should still be able to access the marketplace. It will show up on the top right corner that they are not logged in/login is shown instead of logout.
43. When fetching data from API routes, there should be a maximum response time of 1s. Anything more than this will impede upon user experience.
44. When a user sends an axios request, the server should be able to parse through urlencoded or json data.

Coding Best Practices

45. React components should be kept simple. Design a React component so that it accomplishes one job for the user. If a component is more complex, then use multiple react components with parent/child relationships.
46. An object titled INITIAL_STATE should be used in each Redux reducer to hold each reducer's state data.
47. Redux actions should start with the name of the reducer in capitals, an underscore, plus whatever the action is doing, an underscore, plus whatever the action is changing. Example: USER_UPDATE_POST, USER_SEND_MESSAGE, etc.
48. If a team member must create a new React route, then do so in App.js. This is where all the Routing takes place.

49. Variable names should not be redundant, but instead describe what role the variable has. Thinking of good naming conventions is important so everyone else on the team knows why and how the variable was used.
    a. By default, use camel case for variable names.
50. Commenting is essential. Before jumping into code, every team member should write some form of pseudocode describing what it is they are going to code before actually doing so.
    a. Outside of pseudocode, commenting should be used everywhere to describe how features or architecture works.

Product Constraints

51. At least 3 images of products required on listing. Having more than 1 image of the product provides customers with more information of how the product will look.
52. Minimum/maximum size of images on product page. An appropriate size of the images (perhaps 600x600 or 2x2) will help users with product visibility.
53. At least a 10-word description of the product being sold. The seller will provide adequate details on the product listing.
54. Buyer protections to provide buyers with peace of mind when using the app. (PayPal has 180 days for user to be refunded if there is an issue with the purchased item)
55. At least a 5-word title of the product being sold. The seller will provide an adequate title on the product listing.
56. Once an item is sold, the listing is automatically updated to reflect as such. This will notify users that the listing is no longer available/has been sold, or that the number of products available is now different.
57. Products must be concrete, not abstract (no services, only physical items.) Restrictions for what can be bought and sold on the app should be made clear.
58. Maximum number of listings per seller- 100? 1000? To keep balance of how we are trying to cater to the smaller businesses.
59. Offer commitment on each listing that has the offer option. Once the buyer makes an offer or bid and wins, a transaction automatically occurs, and the buyer gets their account charged (automatic withdrawal).

60. Product page must include how much of each item is available. The number of products available lets the buyer know how much is in stock.
61. Product page has an expiration of 60 days. The expiration ensures that the product listings are current and that sellers are active.
62. Sellers should not be able to reject buyer's meetup time within 2 days unless it is an emergency case.
63. Product page must include a place of origin. This requirement will let the buyer know where the product is coming from, and perhaps what to expect when it comes to shipping time.
64. Product page must include if the item is new or used. This detail is essential to let the buyer know the condition of the product being sold.
65. 24-hour window to modify order; after the 24 hours, buyers will be committed to their order.
66. Must show the seller's name of a selling item.
67. Must show the seller's rating selling items.

Marketing and Revenue

68. Stay connected- follow us on social media. This will provide the app with cross platform exposure and potential business.
69. App news/announcements. Users will have the option to subscribe to the app to receive newsletters through email.
70. Paid ads throughout the site. Advertisements will be another source of revenue for the app. Perhaps the end of the page will include a link titled "advertise with us" to attract potential business.
71. FAQ page that highlights all the questions and answers that users generally have when using the app.
72. Coupon/promo code expiration. Every code will have an expiration date to keep discounts as special.
73. Recently sold items on the home page. This will show users what has recently sold, and for how much it has sold for.

Configuration

74. Team members should use pm2 as their process manager when running the project locally. ‘npm start’ is NOT necessary for running the project. Only ‘pm2 start process.config.js’ should be used.
75. Freenom website should be used to configure a free domain name for our app.
76. If the AWS EC2 instance is stopped and then reset, the public ipv4 address will have changed. This means the app’s public IP address must be changed in /credentials, and it must be updated in freenom’s domain name configuration.
77. Team members should use the ‘systemctl status nginx’ command to test the status of the nginx web server.
78. If necessary, team members should only update the nginx server blocks contained in /etc/nginx/sites-available/default. Nowhere else.
79. In order to verify any errors related to nginx, always use the command ‘sudo nginx -t’
80. If team members need to reload the nginx server, then run ‘sudo systemctl reload nginx’

Git

81. If team members pull changes to a remote branch, make sure to ‘npm install’ so you have any new packages/libraries installed from other people’s commits.
82. Team members should always create new branches from the development branch.
83. After you are done with your work, create a merge request from your branch back into the development branch so we can compare the differences between the two.
84. If you ever have a question about any new code pushed to GitHub, then team members must ask about what they are unsure on in the discord #help channel.
85. When team members encounter a bug in the code, team members should write down the steps to duplicate the bug, then create a pull request labeled as ‘bug’ so that the rest of the time can replicate the bug.
86. If team members need to add additional documentation for a feature they have created, they should add the documentation label onto their pull request.
87. If a team member is creating a new feature, then they should mark their pull request with the enhancement label.

88. If a team member has a question about a portion of code they are writing, then they should create a pull request from their branch back into development with the question label. This way, the rest of the team can review what the question is to resolve it.
89. Code should be merged from the development branch into master branch only when all merge requests for the current build have been agreed upon and added to the current git history.
90. Whenever a milestone is close to submission, a thorough check of the /credentials folder should be done to ensure everyone has the most recent updates.

Remote EC2 Instance

91. SSH should always be used when logging into the remote EC2 instance.
92. Whenever logging into the EC2 instance, make sure to check for updates: use the following commands: 'sudo apt update', 'sudo apt upgrade' or 'sudo apt full-upgrade', and 'sudo apt autoremove'. If you are prompted to reboot, then reboot the instance.
93. Create a CI/CD pipeline for deploying code to the remote ec2 instance.
94. The Node api server should have access to react build files so it works correctly on the remote ec2 instance.
95. In order to login to the EC2 instance correctly, team members should ensure that the csc648.cer file in their local github repository's /credential folder has file permission 600.

UI/UX Constraints

96. CSS styling to create a box-shadow around an item when the user hovers over it. This will allow the user to view the item clearly.
97. High contrast between text and background for ease of use. This will make the overall appearance of the app easy to view.
98. The sidenav should be implemented with the react transition group library
99. All Contact, About, Careers, company information, etc. should be stored in the footer of the app
100. All Login/Logout, Shopping Cart, Notification, Auctions pages should be stored in the sidenav menu
101. @media queries should be used for making the app usable on mobile devices.

102. In order to satisfy requirements for screen sizes <= iPhone 5, the minimum width for @media should be 300px
103. In order to satisfy requirements for screen sizes <= iPhone 6, the minimum width for @media should be 360px
104. In order to satisfy requirements for screen sizes <= Ipad, the minimum width for @media should be 720px
105. In order to satisfy requirements for screen sizes <= Desktop, the minimum width for @media should be 1280px
106. By default, buyers will not have their shipment information added. If they choose delivery, then they must be prompted for their address information.
107. Users should be able to choose what page route that he or she needs to go to such as Home, Notification, About, Contact, Login, and Logout.
108. A hamburger menu should be used to represent the navigation bar, this menu should have an animation which opens and closes it.
109. Users should be able to interact with a “See More” at the end of the top 50 posts, which when clicked will load another 50 posts to view on the Home page.
110. When users click the hamburger menu, there should be a navigation bar component which is triggered in addition to the side nav which should pop out from the side.
111. Navigation bar should have an animation triggered when it is opened or closed.
    a. This task should be implemented with some sort of CSS transform or the use of a React component to represent the navigation bar
112. When a user clicks the Notifications tab in the navigation bar, a dropdown menu should appear.
    a. A list of the user’s notifications ordered by most recent to least recent is a requirement.
113. When a user clicks on a notification from another user, they should be redirected to a page displaying their conversation history.
114. If a user is not logged in, views an item, and then clicks the button to message the seller, the user should be prompted to either login with their username/password or to create an account with a button.
115. Lock password input in login form from being updated or clicked on after the user clicks login.
116. If a user attempts to login and they use the wrong credentials, they should be redirected back to /login route.

117. Once a user logs in with the correct username/password, redirect them to the home page at route '/' to start interacting with our app's features.
118. Users can switch to dark or white background mode for the app.
    a. This helps for those who want to view the app in a different light or contrast.
119. Category menu (beauty/cosmetics, home appliances, tech gadgets/computers, athletic tools, clothing. This will allow searching items faster, and to keep all the products organized.
120. Photo slider which has left/right arrows allowing the buyer to navigate through the pictures of an item posted by a seller. This will allow the buyer to see the details on the design of the product.
121. Have an option to slide through the posts or go to the next page.
122. Display seller's contact information
123. Shopping cart with number of items icon. The buyer will be able to view the total items in their cart and the total cost of the items.
124. Option to drag item(s) to the shopping cart button. This will provide shopping ease for buyers.

6. Competitive analysis

Company	**Facebook Marketplace**	**Craigslist**	**LetGo**	**OfferUp**	**Ebay**	**Mercari**
URL	https://www.facebook.com/marketplace	https://www.craigslist.org/	https://we.letgo.com/	https://offerup.com/	https://www.ebay.com/	https://www.mercari.com/
Strengths	best digital marketplace in current market	oldest digital marketplace, simple business model, easy to use app	free-to-sell model, image recognition and AI capabilities	fun to use app (designed like social media platforms)	most trusted, Auction	Strong Customer support team
Weaknesses	no price-bidding/auction, long chat with buyers to close the deal	scam/fraud deals, outdated platform	requires in-app purchases for boosting listed item and other advanced features	promote and bump paid features for product boost	not preferred for local sales	high fees leave seller with no profit, sellers don't get money until buyer give reviews
Social Media	Facebook, instagram, Twitter, Linkedin, Blogs	Blogs	Facebook, instagram, twitter, youtube,Blogs	Pinterest, facebook, instagram, youtube, linkedin, Blogs	Facebook, twitter, linkedin, Blogs	Facebook, Twitter, Youtube, Instagram, Linkedin, Blogs

Release Year	2016	1995	2015	2011	1995	2013
Pricing	FREE	FREE excluding some services like job, apartment, gifts, furniture, services	FREE (raise funds from investors)	12.9%	12%	10%
Onboarding Experience	Guided step-by-step instructions	Not much support	Functions oriented Onboarding	Seamless, really smooth process	Comfortable buying and selling guide	Good onboarding video in the app
Shipping	5% extra checkout charge	not available	not available	9.9% service fee + shipping fee	10% on final price	based on package weight
User Interface	Similar to facebook	outdated	good	simple mobile-friendly UI	Last update was not up to the marks	good

Features Implemented

Not implemented: -

Implemented: +

Superior feature: ++

Feature	**Facebook Marketplace**	**Craigslist**	**LetGo**	**OfferUp**	**Ebay**	**Mercari**	**Jose's Angels**
Auction	-	-	-	-	++	-	+
In-App Chat	+	-	+	+	+	+	+
Delivery Options	-	-	-	+	+	+	+
Price Matching	-	-	-	-	-	-	++
Daily Deals	+	-	-	-	+	+	+
Advanced Search Bar	+	-	-	-	+	-	+
Discount Coupons	++	+	++	+	+	+	+
Seller Profile Rating	+	-	+	++	+	++	++

Summary of Competitive Analysis

Dropsell is a digital marketplace where a person can create an account as a buyer, seller, or both. Sellers will be able to sell unused or used items locally or globally. Unlike the main competitors in the current market, Dropsell supports local pickups as well as shipping of the products to expand the buyer base.

Our competitors are Facebook Marketplace, Craigslist, OfferUp, Mercari, LetGo and eBay. All of these marketplaces are successful in their business strategies and leading the market in one way or another. Our app's priority is to expand the buyer base by giving delivery options, increasing buyer and seller security via secure payment gateway and in-app chat, and build trust by making seller profiles which store the history of all the listings made by that seller. The app will be beneficial for both buyers and sellers. For buyers, it will provide "Price Matching" through comparison algorithms which will compare products with similar products on Dropsell as well as other competitors like Facebook Marketplace, Craigslist, Offerup products.

For sellers, Dropsell will provide the "Auction" feature which will enable them to set an initial bidding price for the product and the time duration of auction. This feature will add benefits on the seller's side by helping them make more money through sale. The sellers will also have their separate profiles which will contain ratings, reviews and history of listed items.

Dropsell app will solve the current problem of choosing whether to sell products locally or globally. The app will also calculate estimated shipping charges for products. The status regarding in stock items, already sold items and Out of Stock items will get updated runtime. Users will be able to modify their order in a 24-hour window. Today's main threat, which is fake products, will be removed through our app, as it will implement a flagging system which will give tickets to bogus products. Angel app will be cost efficient and will have easy to use and attractive UI which will take least time in searching for best deals online.

7. High-level system architecture and technologies used

API Services

1. UPS developer kit API (subject to change if we go with another delivery service)
2. Stripe.js or Paypal Developer SDK for satisfying checkout experience

List of Backend technologies/frameworks/tools

1. Node.js, v14.17.0
2. Express.js, v4.17.1, for routing only
3. AWS RDS mySQL database, v8.0.23
4. MySQL workbench, v6.3.8
5. AWS EC2 instance 1vCPU with 1 GB RAM
6. Ubuntu 18.04 Server
7. Nginx web server, v1.21.0
8. NPM, v6.14.13
9. Mysql2, v2.2.5
10. Multer for handling images, v1.4.2
11. Websocket, v7.4.6

List of Frontend technologies/frameworks/tools

1. React.js, v17.0.2
2. Axios for server requests, v0.21.1
3. React-Redux for state management, v7.2.4
4. Redux-Thunk library for making redux updates to state easier, v2.3.0
5. Browser support for Chrome/Firefox/Safari/IE
6. CSS

8. Checklist

Task	Status
Team found a time slot to meet outside of class	ON TRACK ➔ Meeting 06/04/2021, 4:00pm PST 7:00pm EST ➔ Meeting 06/08/2021, 2:30pm PST 5:30pm EST ➔ Meeting 06/11/2021, 7:00pm PST 10:00pm EST ➔ Meeting 06/14/2021, 6:30pm PST 9:30pm EST ➔ Meeting 06/15/2021, 7:00pm PST 10:00pm EST ➔ Meeting 06/17/2021, 7:00pm PST 10:00pm EST ➔ Meeting 06/18/2021, 7:00pm PST 10:00pm EST ➔ Meeting 06/29/2021, 8:30pm PST 11:30 pm EST
Github master chosen	DONE
Team decided and agreed together on using the listed SW tools and deployment server	DONE
Team ready and able to use the chosen back and frontend frameworks and those who need to learn are working on learning and practicing	DONE
Team lead ensured that all team members read the final M1 and agree/understand it before submission	DONE
Github organized as discussed in class (e.g. master branch, development branch, milestone documents, credentials folders)	DONE

9. List of team contributions

Student Name	Contributions
Mitchel Baker	Functional and non-functional requirements, high-level system architecture and technologies used, checklist, and helped to better organize our list of main data items and entities
Charmaine Eusebio	Functional and non-functional requirements, main use cases, general organization
Kenneth N Chuson	List of main data items and entities, functional and nonfunctional requirements
Krina Bharatbhai Patel	Competitive Analysis, M-1 Editor, Functional requirements
Michael Schroeder	Main use cases, use case diagrams, nonfunctional requirements
Rowena Elaine Echevarria	Functional and non-functional requirements, organization, Competitive Analysis
Jamie Dominic Walker	Executive summary, Functional requirements

M2

SW Engineering CSC648-848 Summer 2021
dropsell.gq, Jose's Angels

Team 03

Name	Email	Roles
Mitchel Baker	mbaker3@mail.sfsu.edu	Team lead
Krina Bharatbhai Patel	kpatel11@sfsu.edu	Frontend lead
Charmaine Eusebio	ceusebio1@mail.sfsu.edu	Frontend engineer
Rowena Elaine Echevarria	rechevarria@mail.sfsu.edu	Frontend engineer
Michael Schroeder	mschroeder@mail.sfsu.edu	Backend lead, Github master
Kenneth N Chuson	kchuson@mail.sfsu.edu	Backend engineer
Jamie Dominic Walker	jwalker5@mail.sfsu.edu	Backend engineer

Milestone 2
July 8, 2021

History Table

Version	Date	Notes
M1V1	06/22/2021	
M1V2	07/02/2021	
M2V1	07/08/2021	
M2V2	07/20/2021	

1. Data Definitions

1. Unregistered User: A user who can visit the website, explore the products displayed in the marketplace, and checkout any public facing pages. Unregistered users need to register to use all the features of the website related to viewing auctions, commenting and rating products, messaging buyers/sellers, and purchasing products.
    a. A general user shall be able to create an account.
    b. A general user shall choose the option of creating an account as buyer/seller/both.

2. Registered User: A registered user is able to access the website with all features. Registered users need to login to the website for buying or selling products.
    a. User login
        i. username
            1. Must enter username to login
        ii. password
            1. Must enter a valid password, encrypted with Bcrypt library

    b. User account details
        i. user_id
        ii. email
        iii. first_name
        iv. last_name
        v. phone
        vi. street
        vii. city
        viii. state
        ix. zip
        x. created_at
        xi. is_active
        xii. Payment details
            1. Attributes
                a. CC number
                b. Expiration date
                c. 3 digit code
                d. Zip
            2. Payment details should be handled by Stripe.js. We may store payment details in the database for multiple payment options in the future.
        xiii. User is a buyer? Seller?
            1. is_buyer
            2. is_seller

    c. User conversations
        i. conversation_id
        ii. sending_user_id (FK to sending user)
        iii. receiving_user_id (FK to receiving user)
        iv. Messages
            1. message_id
            2. conversation_id (FK to conversation id)
            3. message_timestamp

   d. User meetups
      i. meetup_id
      ii. buyer_id (FK to buyer id)
      iii. seller_id (FK to seller id)
      iv. meetup_time
      v. meetup_location

3. User session
   a. Users should have an active session created in order to keep track of the date and time they logged in. This information will be used to keep track of the length of the user's session.
   b. When the user logs in, their session data should be updated in the database.
   c. If the user's session has expired, then the user should be required to log in again.
   d. If, for example, the user's session is still active and they decide to refresh the page, then the user should stay logged in.
   e. Attributes
      i. session_id
      ii. session_expires
      iii. session_data

4. Buyers: can browse products and buy them.
   a. shopping_cart
      i. shopping_cart_id
      ii. buyer_id (FK to buyer id)
      iii. subtotal
      iv. shopping_cart_products
         1. product_id
         2. shopping_cart_id (FK to shopping cart id)
         3. title
         4. price
         5. quantity

5. Sellers: can upload product information and sell them.
   a. Seller Ratings
      i. seller_rating_id
      ii. seller_id
      iii. seller_rating (5-star rating system, from 1-5)

   b. Products: The items which are uploaded by sellers, and purchased by buyers.
      i. product_id
      ii. seller_id
      iii. title
         1. At Least 5 word Title or name of the product
      iv. description
         1. At Least 10 word Description of the product
      v. image
         1. At least 1 image with size of (600x600 or 2x2) for product visibility
      vi. price
         1. The price for product (For sale and auction only)
      vii. category

c. Product Comments
    i. product_comment_id
    ii. product_id (FK to product id)
    iii. creator_id (FK to user id)
    iv. comment_timestamp
    v. comment

d. Product Ratings
    i. product_rating_id
    ii. product_id (FK to product id)
    iii. creator_id (FK to user id)
    iv. product_rating (5-star rating system, from 1-5)

e. Product Refunds
    i. product_refund_id
    ii. product_id (FK to product id)
    iii. buyer_id (FK to buyer id)
    iv. seller_id (FK to seller id)
    v. refund_amount

6. Auction Products
    a. product_id
    b. seller_id (FK to seller id)
    c. starting_bid
    d. auction_duration

7. Top-Purchased Products
    a. product_id
    b. seller_id (FK to seller id)
    c. total_purchased
    d. added_at

8. Daily Deal Products
    a. product_id
    b. seller_id (FK to seller id)
    c. deal_duration

9. Shipping Products
    a. product_id
    b. buyer_id (FK to buyer id)
    c. seller_id (FK to seller id)
    d. shipping_from
    e. shipping_to
    f. transaction_total

10. Redux related data definitions
    a. Login
        i. Actions
            1. setUsername(username)
                a. Action type: ‘USER_SET_USERNAME’
            2. setPassword(password)

        a. Action type: ‘USER_SET_PASSWORD’
    3. loginUser()
        a. userData(username, password)
    4. redirectUserAfterLogin(loggedIn)
        a. Action type: ‘USER_IS_LOGGEDIN’
  ii. Reducer
    1. INITIAL_LOGIN_STATE
        a. Username
        b. Password
        c. loggedIn

b. Register
  i. Actions
    1. setUsername(username)
        a. Action type: ‘USER_SET_USERNAME’
    2. setPassword(password)
        a. Action type: ‘USER_SET_PASSWORD’
    3. setConfirmPassword(confirmPassword)
        a. Action type: ‘USER_SET_CONFIRM_PASSWORD’
    4. createUser()
        a. userData(username, password, confirmPassword)
    5. redirectUser(registered)
        a. Action type: ‘USER_IS_REGISTERED’
  ii. Reducer
    1. INITIAL_REGISTER_STATE
        a. Username
        b. Password
        c. confirmPassword
        d. Registered

c. Products
  i. Actions
    1. setTitle(title)
        a. Action type: ‘PRODUCT_SET_TITLE’
    2. setDescription(description)
        a. Action type: ‘PRODUCT_SET_DESCRIPTION’
    3. setPrice(price)
        a. Action type: ‘PRODUCT_SET_PRICE’
    4. setImage(image)
        a. Action type: ‘PRODUCT_SET_IMAGE’
    5. setSuccess(isSuccess)
        a. Action type: ‘PRODUCT_SET_SUCCESS’
    6. setCategory(category)
        a. Action type: ‘PRODUCT_SET_CATEGORY’
    7. setCategories(categories)
        a. Action type: ‘SET_CATEGORIES’
    8. changeDropdownText(text)
        a. Action type: ‘CHANGE_DROPDOWN_TEXT’
    9. createProduct()
        a. formData(title, description, price, category, file)
    10. getProducts(products)

            a. Action type: ‘GET_PRODUCTS’
        ii. Reducer
            1. INITIAL_PRODUCT_STATE
                a. Title
                b. Description
                c. Price
                d. Category
                e. File
                f. filePreview
                g. isSuccess
                h. Products
                i. Categories
                j. dropdownText

11. Search data definitions
    a. Query
    b. searchQuery/setSearchQuery()
    c. filterProducts(products, query)

2. Prioritized Functional Requirements

Priority 1

Marketplace

1. Unregistered and registered users shall be able to query the database for products by interacting with a search bar.
2. Sellers shall be able to put their products up for auction or list them in the marketplace
3. Buyers shall be able to propose buy it now or best offer options for a seller's product
4. Users should be offered a base price for automatic reject or accept
5. Buyers shall be able to access the location information of sellers through an interactive map
6. Unregistered and registered users shall be shown ads of products similar to their search histories
7. Buyers shall be given a clear breakdown of shipping options to choose from
8. Buyers shall be able to check their account settings to determine if a product they've purchased has been confirmed, processed, shipped, or returned
9. Buyers shall be given a tracking number to stay up to date with their product's delivery status
10. Products shall be displayed as out of stock if the supply has run out
11. Buyers shall be contacted by email if they opt to be notified when a product comes back in stock

Website Features

12. Sellers shall be provided with consignment operations to get their products delivered to buyers
13. Buyers shall be given a price matching tool, which compares products either already posted on our app or compares other products from other websites (amazon)
14. Buyers shall be provided with daily deals on the home page
15. Users shall be able to zoom in or out of product images with a magnifying glass feature
16. The checkout page shall calculate tax automatically based on the buyer's location

Buyers

17. Buyers shall sign a purchase agreement; they must agree to conduct business according to Dropsell's rules prior to using the marketplace
18. Buyers shall be able to send inquiries of interest to a product's seller
19. Buyers shall save products they wish to purchase by adding them into a shopping cart
20. Buyers shall be able to remove an item from their shopping cart and the total list of products should be updated accordingly
21. Buyers shall be able to return to the main shopping menu to look for other products if they are not finished shopping
22. Buyers shall have the option of canceling or modifying their order
23. Buyers shall receive a detailed receipt of their purchase through email after checking out

24. Buyers shall receive shipping information through email if their purchase involves shipping products to their specified address
25. Buyers shall be able to add their full name, email, phone number, and delivery address if they haven't inputted this data prior to checking out
26. Buyers shall have the option of choosing an existing payment option or adding a new one
27. Buyers shall be notified by final invoice of their purchase's pickup/delivery time, expected time of arrival, name of seller, location to meet them at, their contact info, and the grand total to pay
28. Buyers shall be able to rate and star products
29. Buyers shall be able to interact with all of their starred products in their buyer settings
30. Buyers shall be able to choose from various UPS delivery methods
31. Buyers shall be able to view all data related to a seller's review page
32. Products shall have their reviews displayed when listed in the marketplace
33. Buyers shall be permitted to submit reviews of products they purchase
34. Buyers shall add auction products to their watch list to keep track of the auction's most recent updates
35. Top-rated products shall be displayed to buyers after they purchase a similar product

Sellers

36. Sellers shall be able to create new products with a title, description, price, category, and image
37. Sellers shall be able to edit the title, description, price, category, and image of their products
38. Sellers shall sign a contract before being granted the privilege of posting products on Dropsell
39. Sellers shall agree to a small listing fee for each product they sell
40. Sellers shall be able to adjust the quantity of a product they've listed on Dropsell
41. Sellers shall be able to delete products they've listed
42. Sellers shall have the option of selecting multiple products to delete
43. Sellers shall be shown all their listed products under the seller settings section of their profile
44. Sellers shall have the option of relisting their products for sale
45. Sellers shall get an email confirmation after one of their products has sold

Auction

46. Buyers shall be able to filter the products they are searching for, based on minimum/maximum price, location filtering, or type of shipping (pickup/delivery)
47. Sellers shall have full control over their auction settings, from choosing the average starting bid, the time when the auction begins, to the total duration of the auction
48. Buyers shall be displayed the current price of an auction, the amount of time remaining, and any bids from other buyers
49. Buyers shall be able to bid on products with one click.
50. Buyers and sellers shall see a live countdown of the auction's remaining time
51. Sellers shall be able to set the duration of their auction for as long as 30 days or as short as 1 hour
52. Buyers shall not be able to retract a bid they place on a product

53. Buyers shall be notified immediately by email that they've won an auction

Messaging

54. Buyers shall have the ability to contact sellers directly through the Dropsell website
55. Buyers shall have the ability to contact sellers via email or phone, if they choose to do so

Priority 2

Marketplace

56. Buyers shall be given an advanced search bar for detailed searches, including product sizes, price ranges, and colors
57. Sellers shall be able to publish different styles of their products, such as different colors or fabrics
58. Buyers and sellers shall be able to schedule and edit meetup times
59. Buyers shall be refunded in full if they purchase multiple products which causes the product to be sold out
60. Unregistered and registered users shall be able to share products with friends through email, social media, or a shareable link
61. Registered users shall have the option of flagging products as inappropriate, a scam, or submit a ticket to the Dropsell team for further review
62. Unregistered and registered users shall be able to go back to products they have viewed previously on their user feed
63. The marketplace shall show the maximum time length for shipping a product
64. The marketplace shall show the minimum time length for shipping a product
65. The marketplace shall perform monitoring on unregistered and registered users
66. The marketplace shall keep track of the ID's of posts unregistered and registered users click on which will be saved in the database under most-interacted products
67. The marketplace shall contain a currency converter for entering international markets

Website Features

68. Sellers shall be provided with a daily product forecast
69. Sellers shall be provided with a profitability test for determining how much it costs to ship a product on Dropsell as opposed to other ecommerce services
70. Sellers shall be provided with an algorithm which keeps track of how many interactions their products have
71. Products shall contain hashtags to help with categorization
72. Unregistered and registered users shall be able to search products using hashtags
73. Buyers shall be shown products which are similar in price or category to the product they are currently viewing
74. Buyers shall be shown comparisons between product specifications
75. Sellers shall be able to see statistics related to how many buyers have purchased their products
76. Sellers shall be able to keep track of the median price ranges of products they are viewing

Buyers

77. Buyers shall be able to click on product listing images to zoom in and see more details
78. Buyers shall be provided with a list of their purchase histories
79. Buyers shall be able to save products they'd like to buy in the future to a wishlist
80. Buyers shall be rewarded with discount codes if they are frequent buyers
81. Buyers shall be provided with a promotion code box for entering their discount code when checking out
82. Buyers shall be able to generate a referral URL which they can send to friends
83. Buyers shall be able to subscribe to a specific seller so they can be notified when a product's price is updated, the product has been restocked, or if the product has been removed
84. Buyers shall be notified about products they have viewed previously
85. Buyers shall be asked if they are still interested in the products they've previously viewed

Sellers

86. Sellers shall allow their data such as user details, products, profile picture, to be publicly displayed to buyers
87. Sellers shall be able to advertise their products to specific buyers with a send offer feature
88. Products shall have a review page where buyers send feedback about its quality
89. Sellers shall have a review page where buyers can provide structured feedback
90. Sellers shall have a rating system based on a five-star system
91. Sellers shall be required to display "illegal item" on illegal products
92. Sellers shall be required to explain their illegal item and their reasons for listing it
93. Seller products shall be marked as out of stock if its supply runs out
94. Seller products shall be marked as "Last 1 Available" so buyers know that it is the last product remaining
95. Sellers shall participate in a striking system if they choose to perform misconduct or violate the terms of service.
96. Sellers shall have 3 strikes before their account faces a possible suspension or blacklist

Auction

97. Buyers shall bid on products as many times as possible before the remaining time runs out
98. Buyers shall be able to keep track of auction statistics
99. Buyers shall be able to see how many other buyers are bidding on the same product
100. Buyers shall be notified with a 5 minute warning before an auction finishes
101. Buyers shall be notified when the auction finishes
102. Buyers shall be notified after they win an auction
103. Buyers shall have the option of choosing from multiple payment options, such as Paypal, ApplePay, or GooglePlay

Priority 3

Marketplace

104. Sellers shall be able to create a wedding registry with a product wishlist for their special day
105. Buyers shall be shown other buyers who have made the same purchases
106. Buyers and sellers shall have the time length of shipping a product hidden if their location is closer than 1 or 2 miles

Buyers

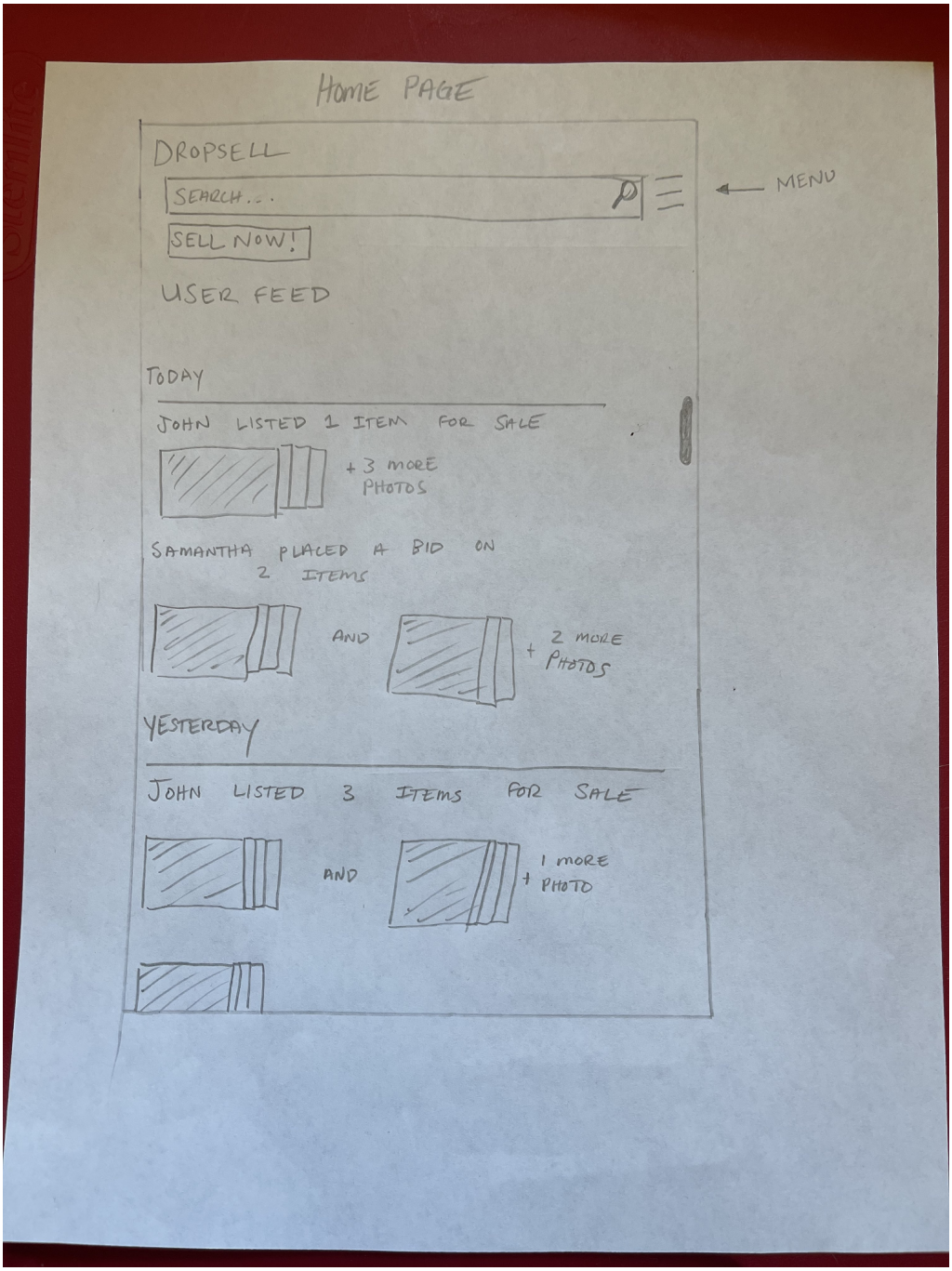
107. Buyers shall be awarded with a random product if they purchase products deemed as lucky
108. Buyers shall be rewarded with a promotion code or a random product for their birthday

Sellers

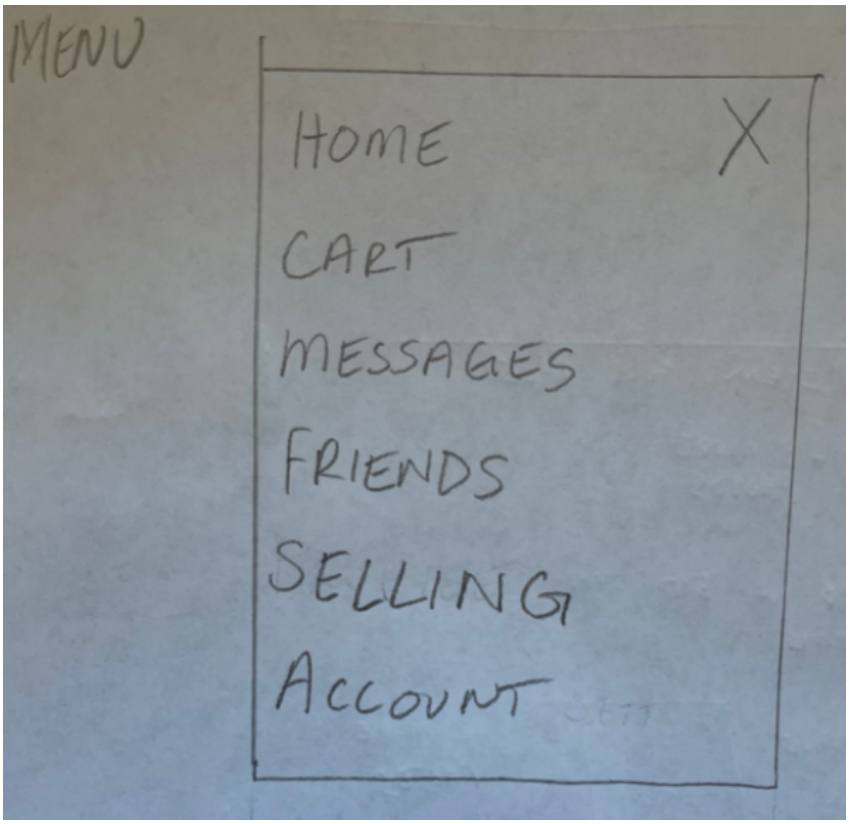
109. Sellers shall have background checks if they intend to sell on Dropsell
110. Sellers shall have their products hide ratings if they have 5 stars

3. UI Mockups and Storyboards

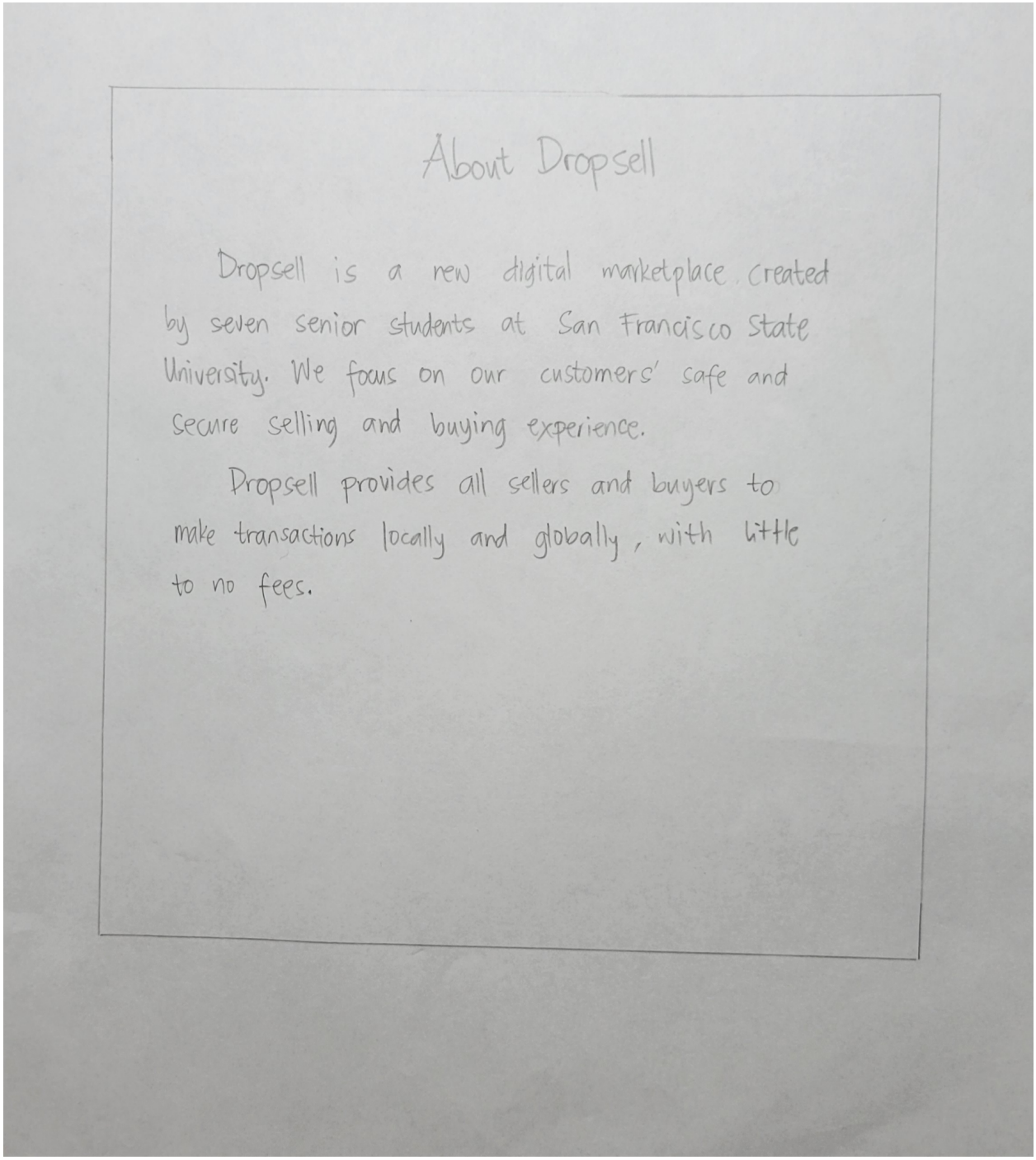
Home Page



Men



About Pag

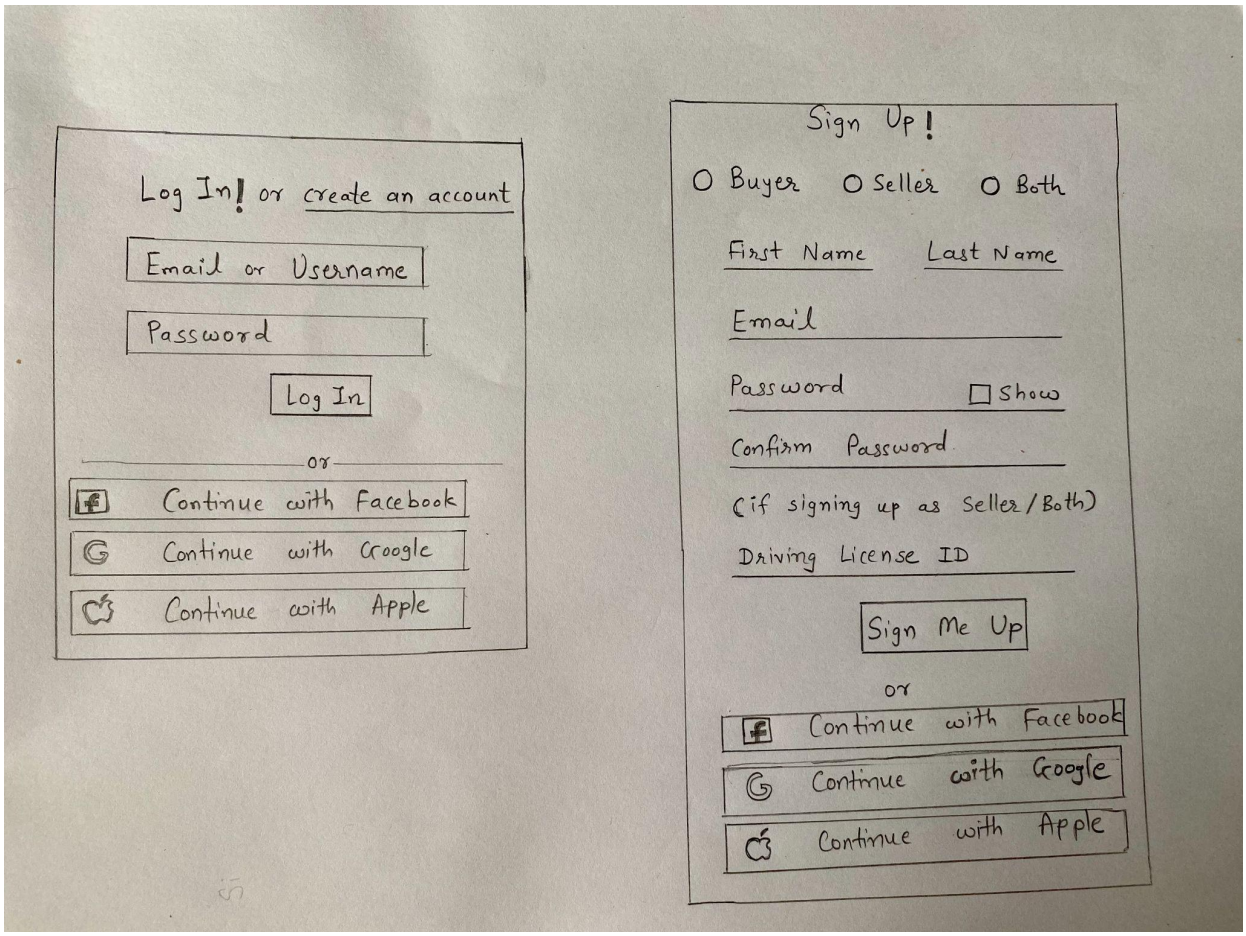


About Dropsell

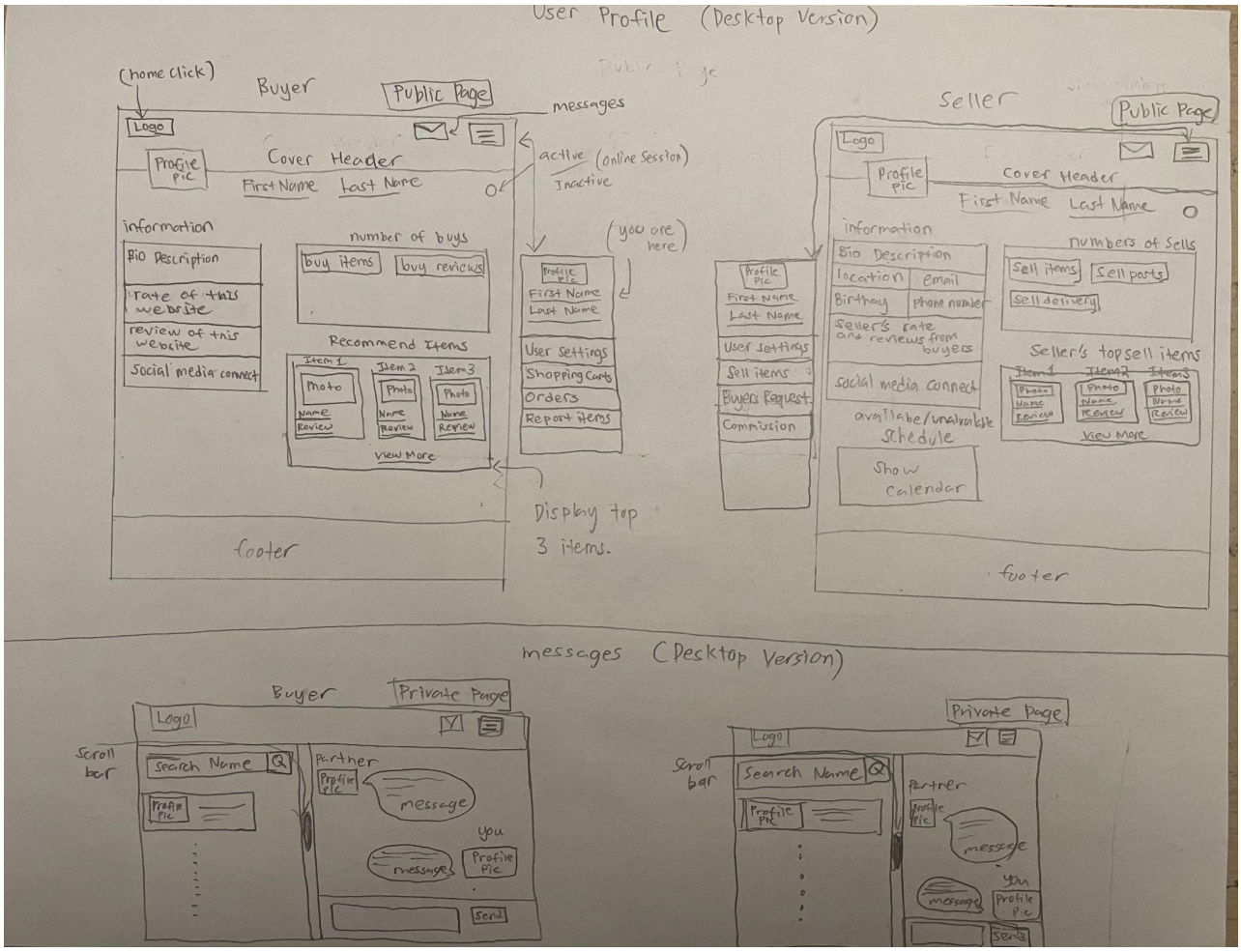
Dropsell is a new digital marketplace created by seven senior students at San Francisco State University. We focus on our customers' safe and secure selling and buying experience.

Dropsell provides all sellers and buyers to make transactions locally and globally, with little to no fees.

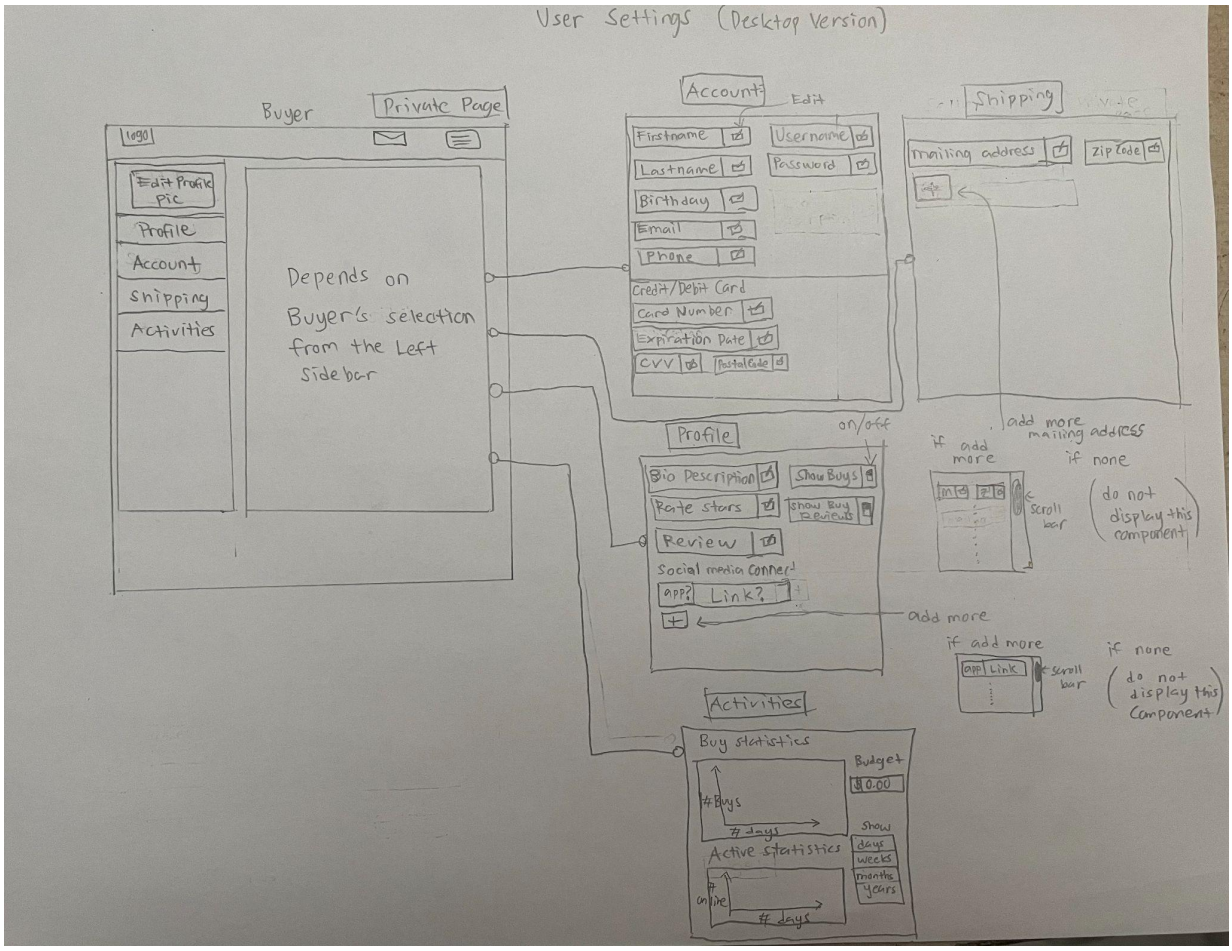
User Sign up / Login



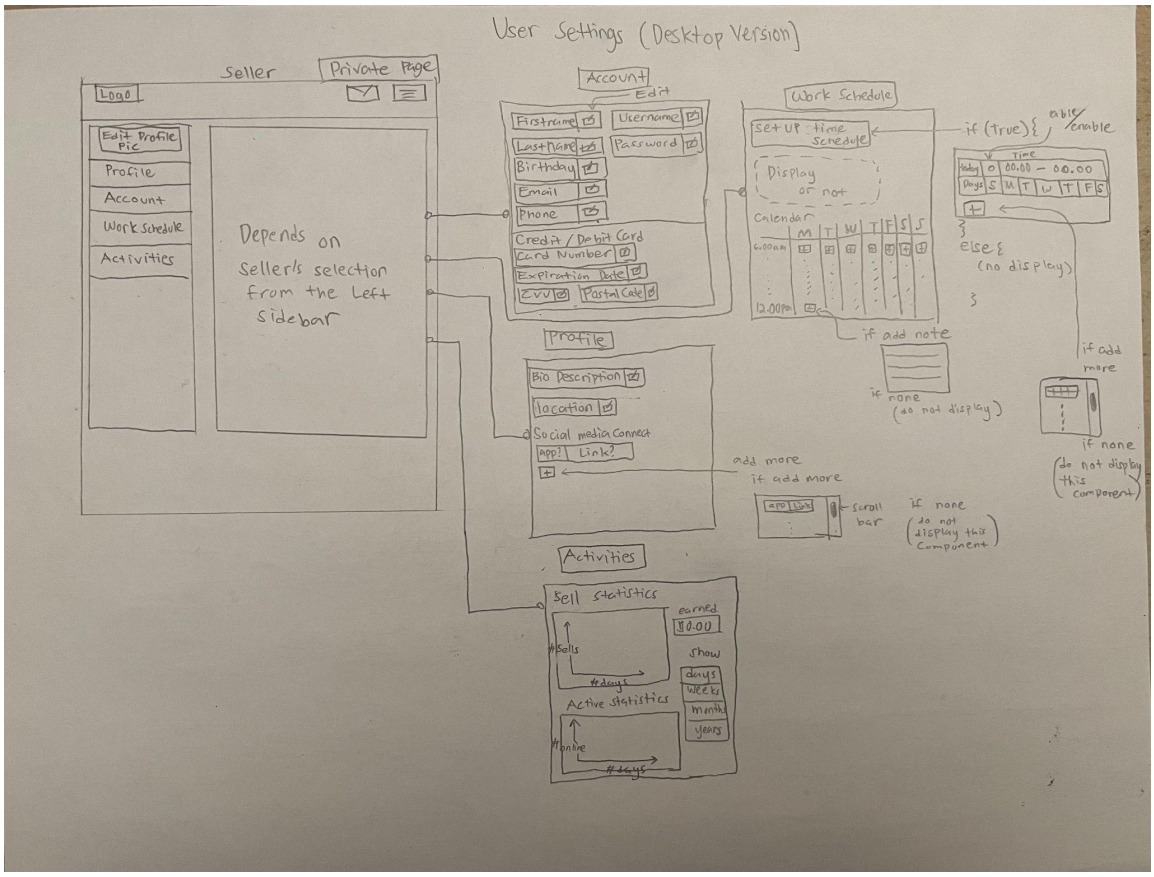
User Profile



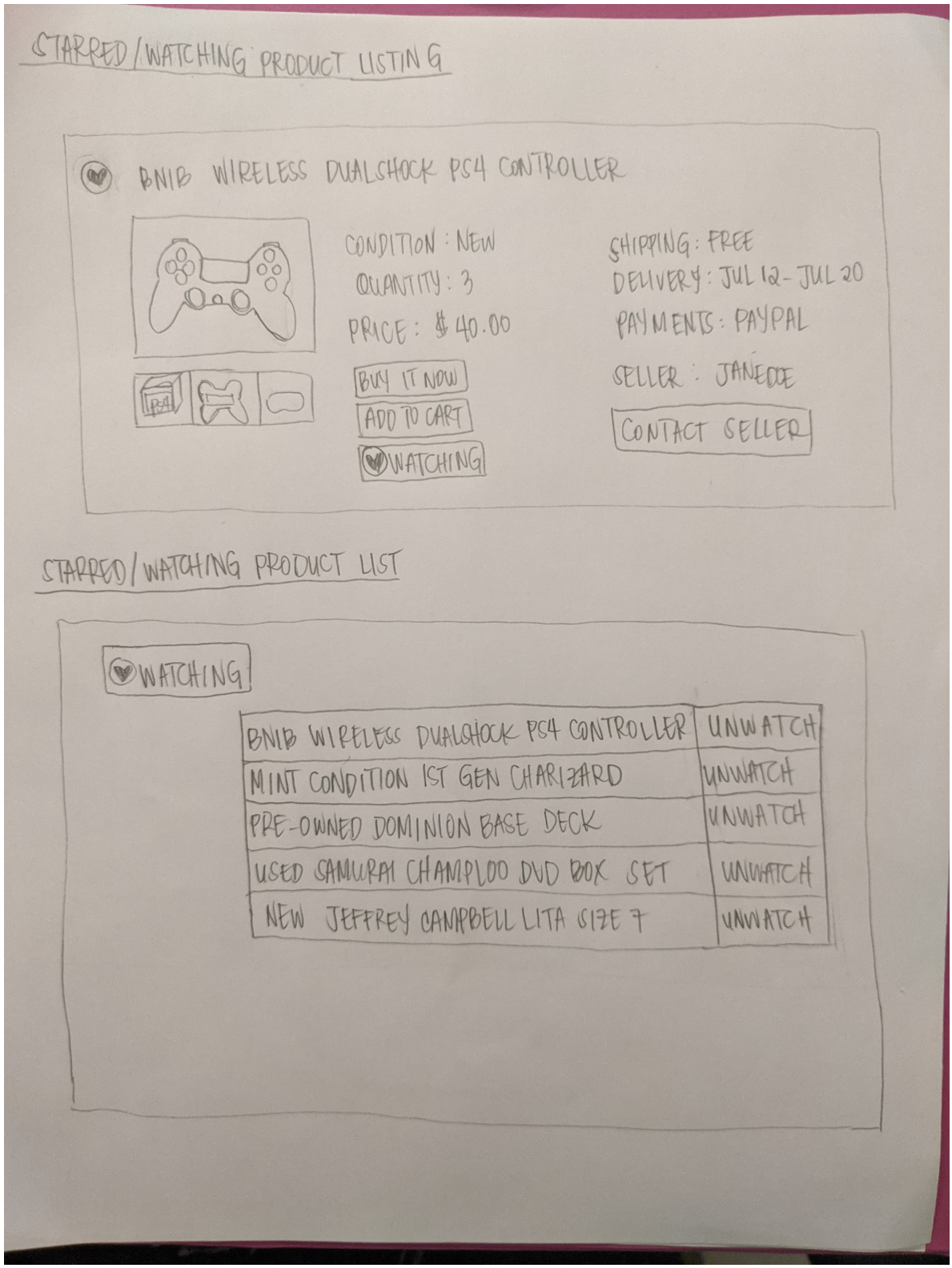
Buyers Settings



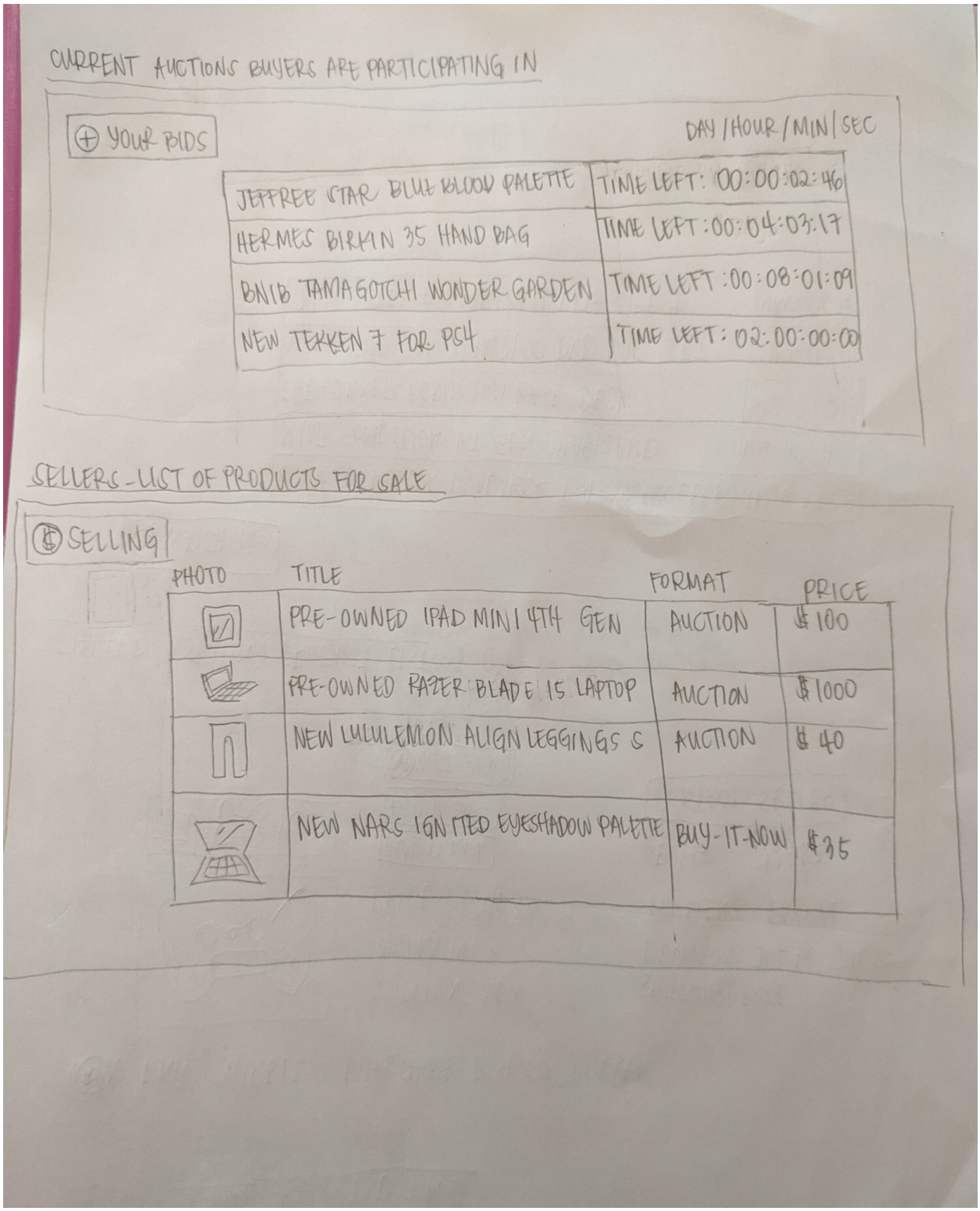
Sellers Settings



Starred/Watching Product Listing, Starred/Watching Product List



List of Auctions and Products for Sale



CURRENT AUCTIONS BUYERS ARE PARTICIPATING IN

⊕ YOUR BIDS

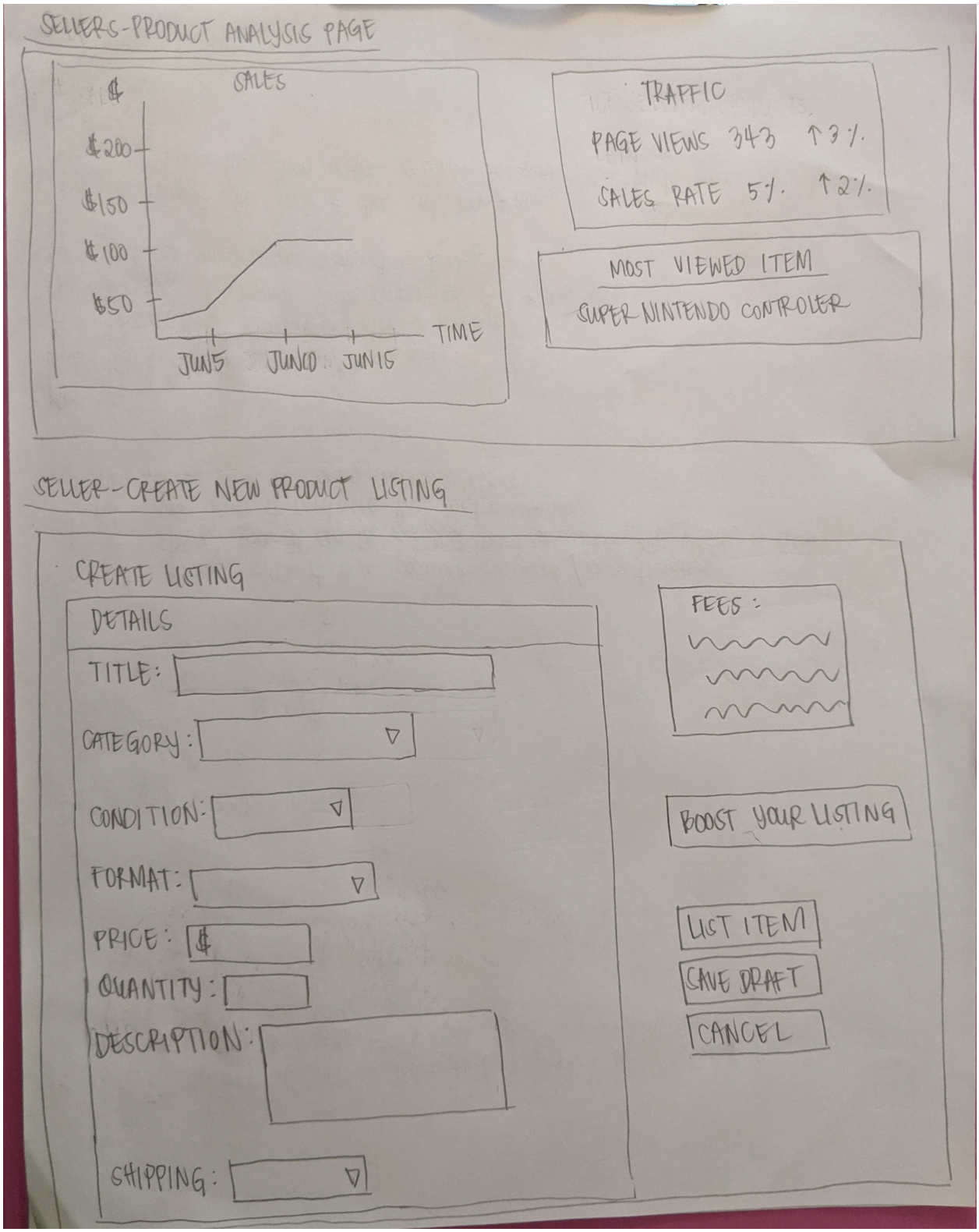
	DAY / HOUR / MIN / SEC
JEFFREE STAR BLUE BLOOD PALETTE	TIME LEFT: 00:00:02:46
HERMES BIRKIN 35 HAND BAG	TIME LEFT: 00:04:03:17
BNIB TAMAGOTCHI WONDER GARDEN	TIME LEFT: 00:08:01:09
NEW TEKKEN 7 FOR PS4	TIME LEFT: 02:00:00:00

SELLERS - LIST OF PRODUCTS FOR SALE

$ SELLING

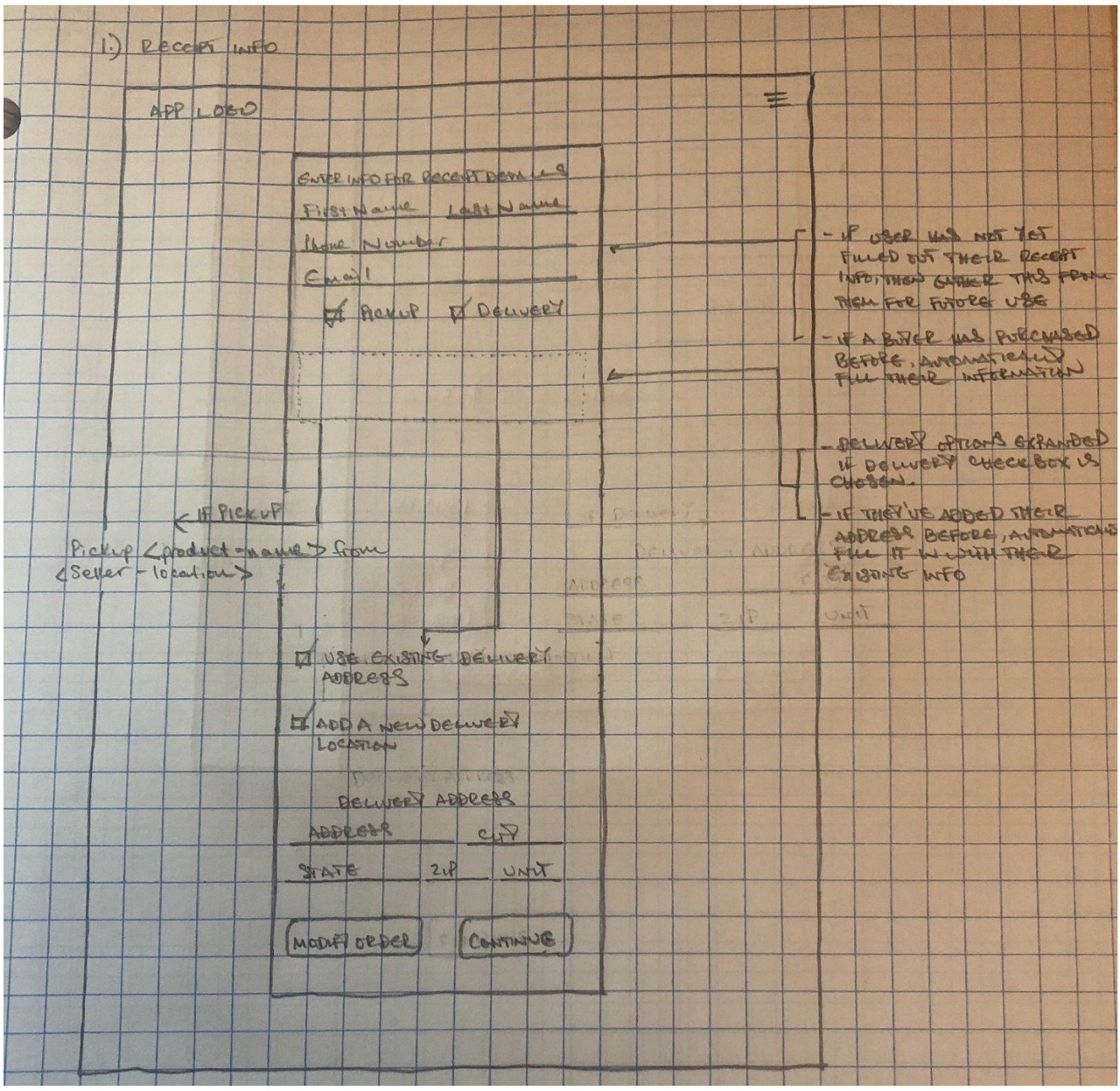
PHOTO	TITLE	FORMAT	PRICE
	PRE-OWNED IPAD MINI 4TH GEN	AUCTION	$100
	PRE-OWNED RAZER BLADE 15 LAPTOP	AUCTION	$1000
	NEW LULULEMON ALIGN LEGGINGS S	AUCTION	$40
	NEW NARS IGNITED EYESHADOW PALETTE	BUY-IT-NOW	$35

Seller Analytics and Create Product Listing Page

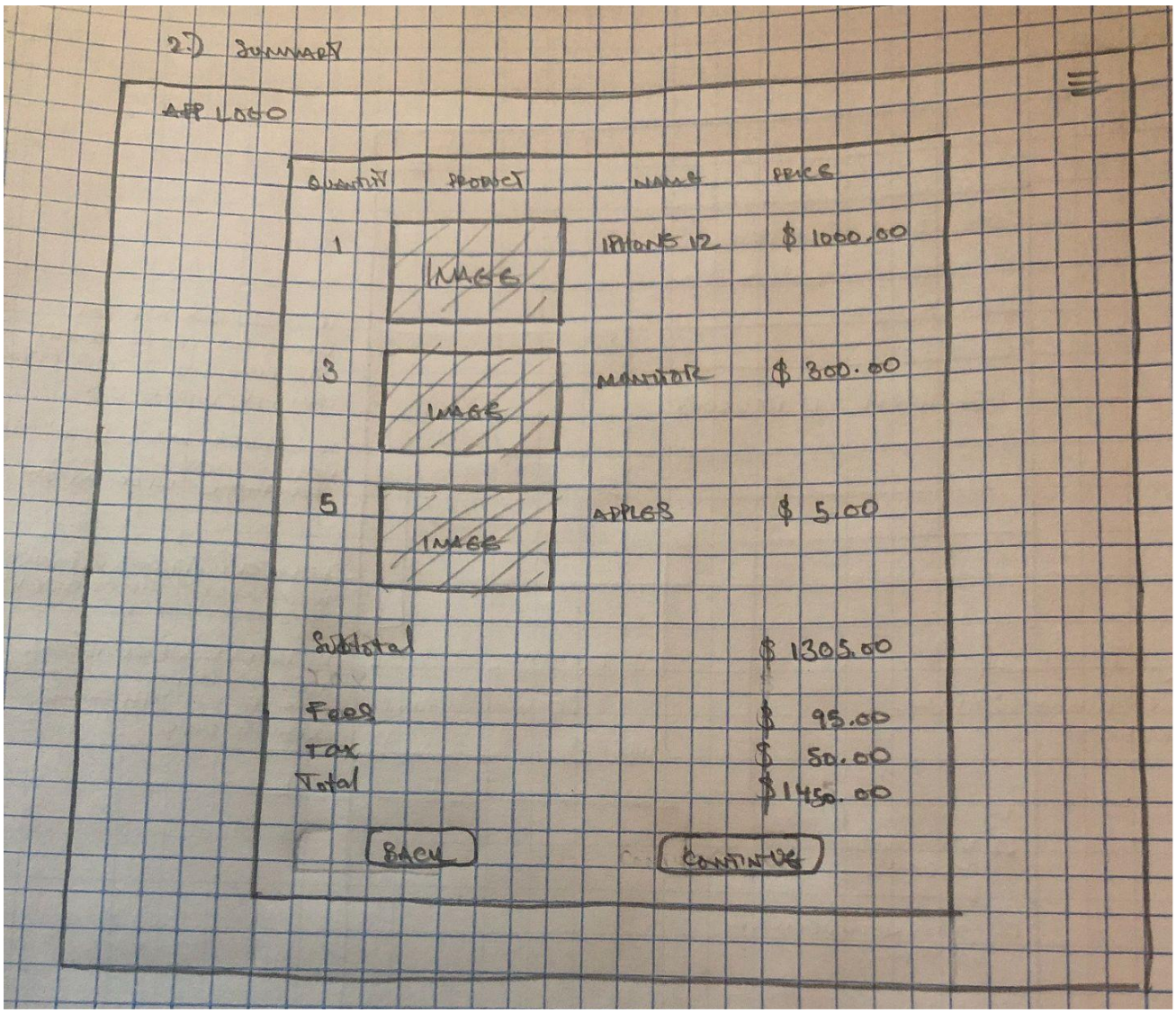


User Checkout Experience

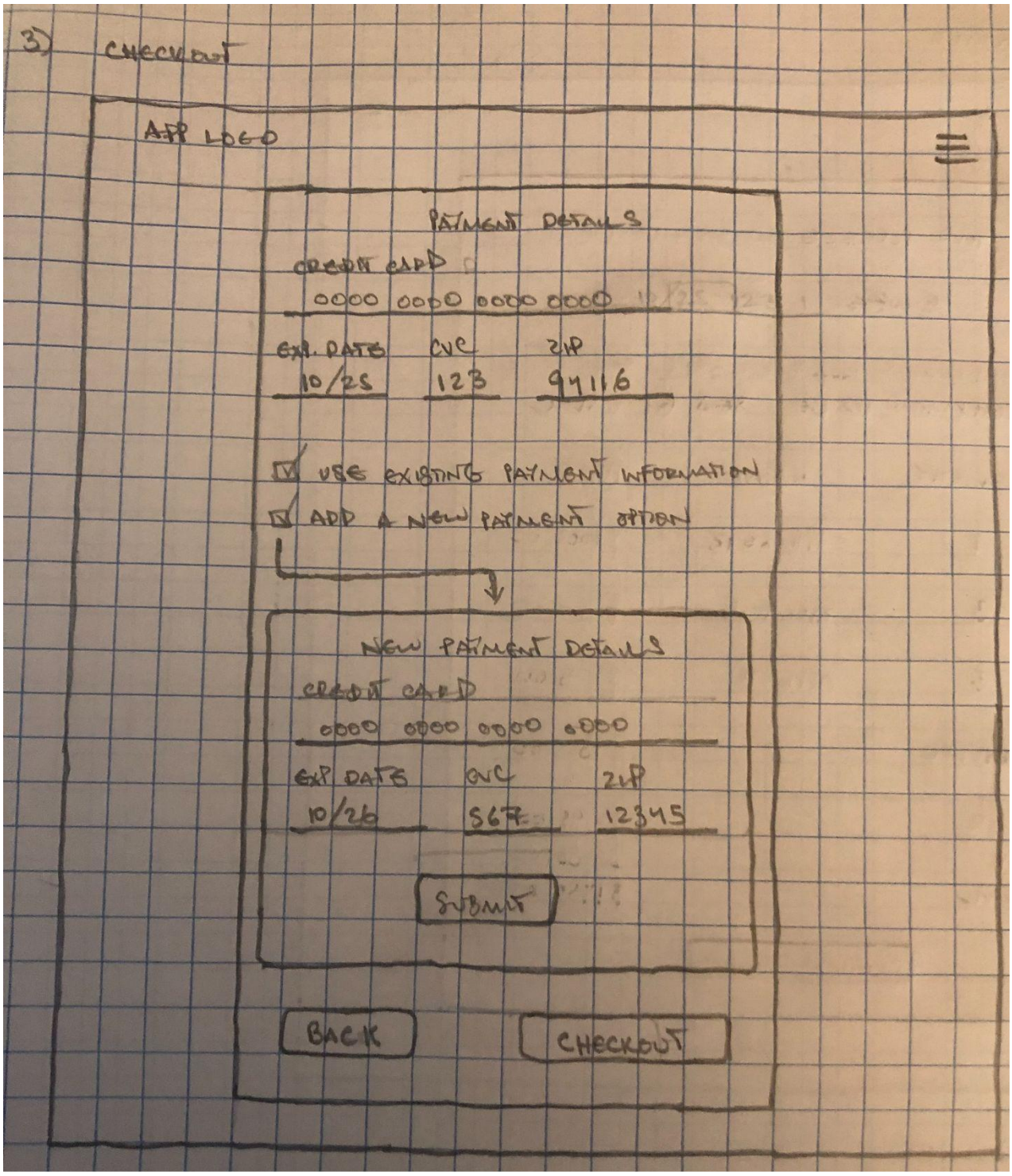
Receipt Info



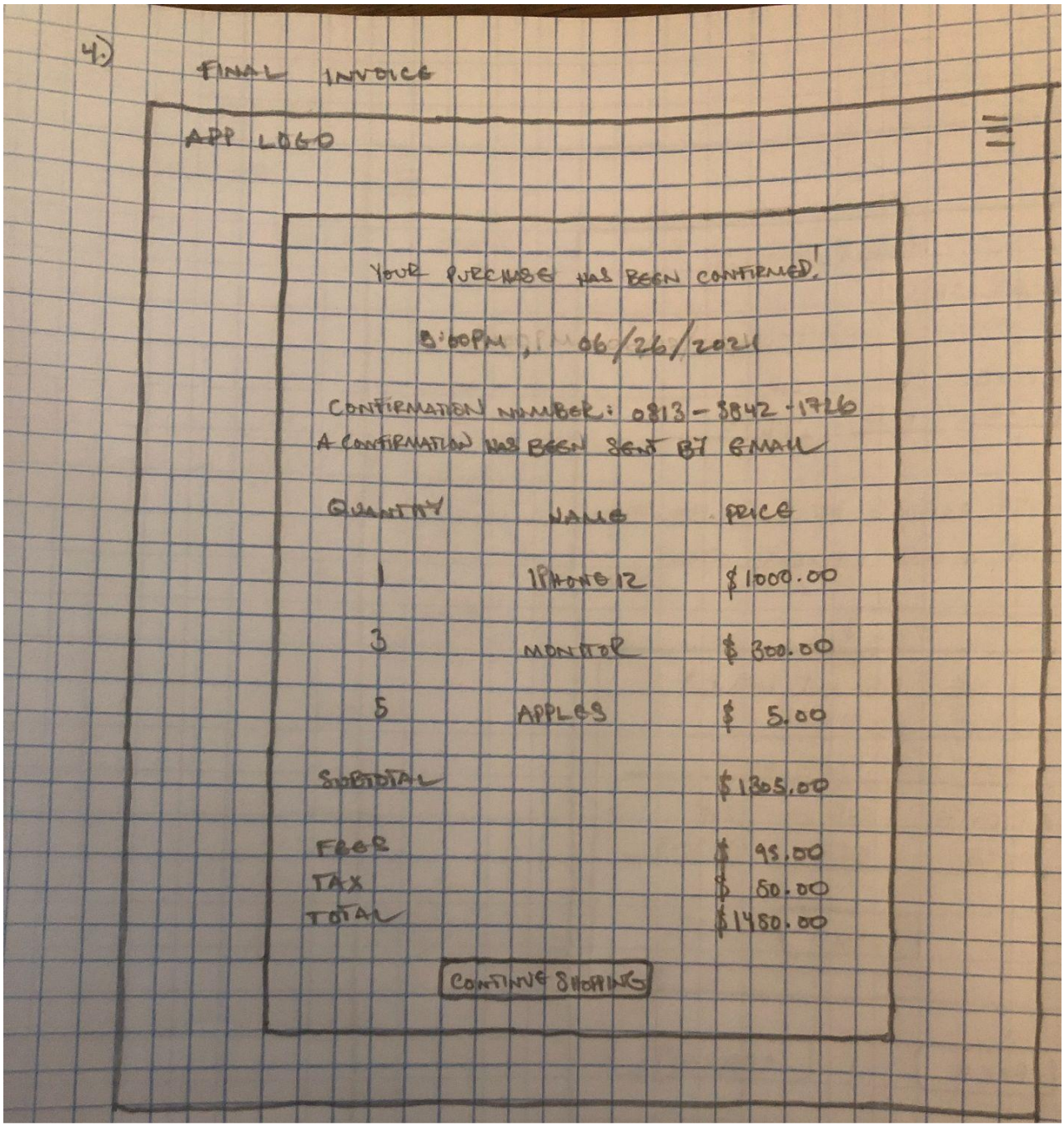
Summary


2.) SUMMARY
APP LOGO
QUANTITY
PRODUCT
NAME
PRICE
1
IMAGE
IPHONE 12
$ 1000.00
3
IMAGE
MONITOR
$ 300.00
5
IMAGE
APPLES
$ 5.00
Subtotal
$ 1305.00
Fees
$ 95.00
Tax
$ 50.00
Total
$ 1450.00
BACK
CONTINUE

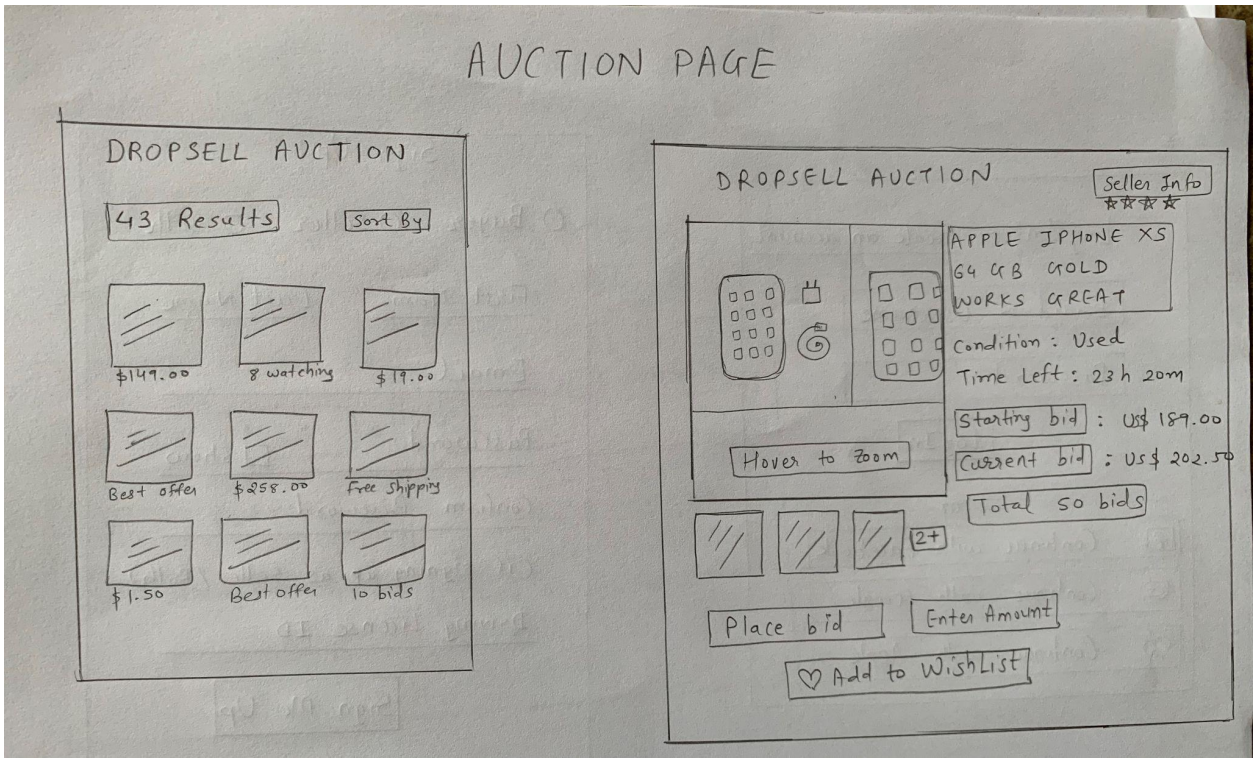
Checkout


3) CHECKOUT
APP LOGO
PAYMENT DETAILS
CREDIT CARD
0000 0000 0000 0000
EXP. DATE
10/25
CVC
123
ZIP
94116
USE EXISTING PAYMENT INFORMATION
ADD A NEW PAYMENT OPTION
NEW PAYMENT DETAILS
CREDIT CARD
0000 0000 0000 0000
EXP DATE
10/26
CVC
567
ZIP
12345
SUBMIT
BACK
CHECKOUT

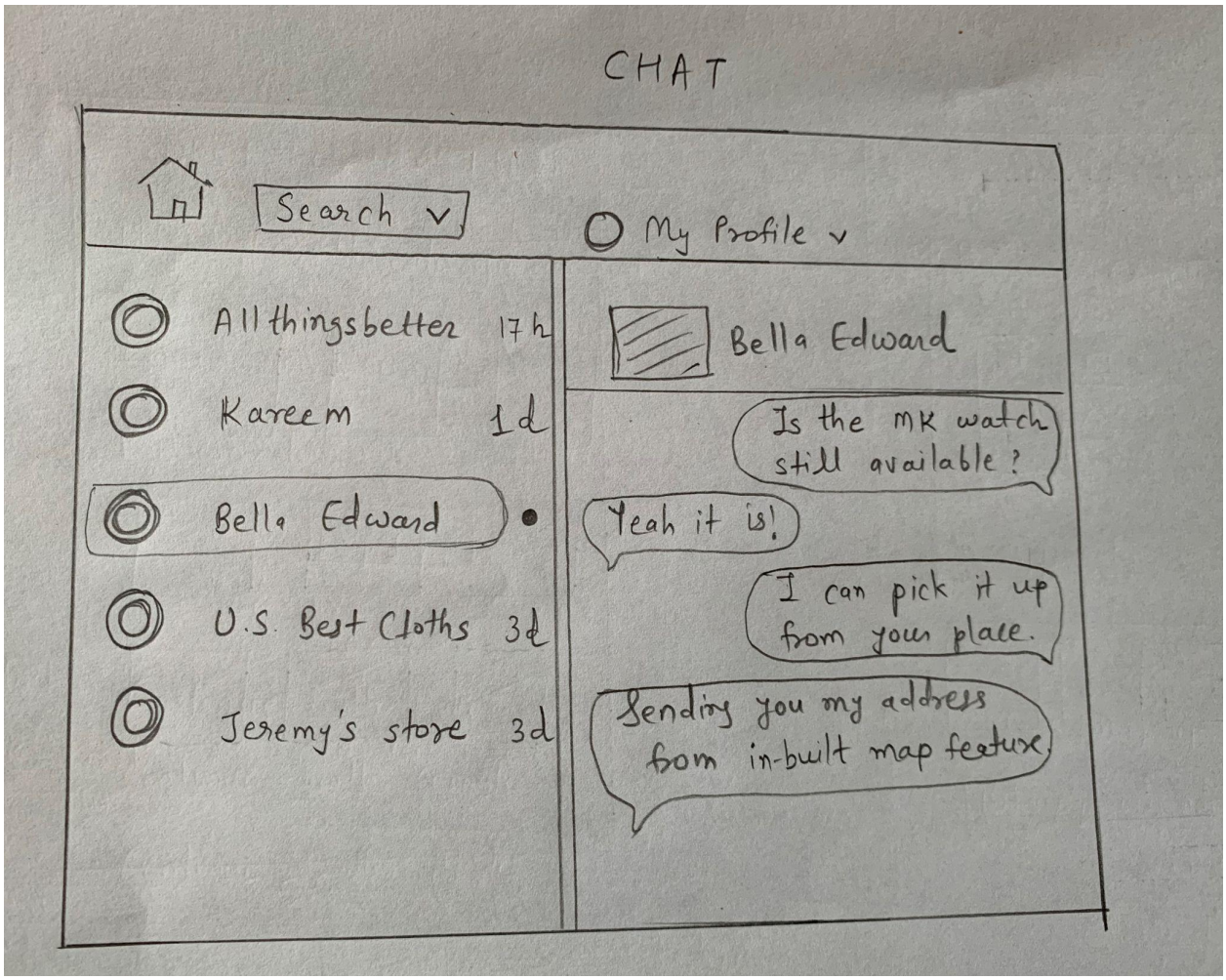
Final Invoice



Auction

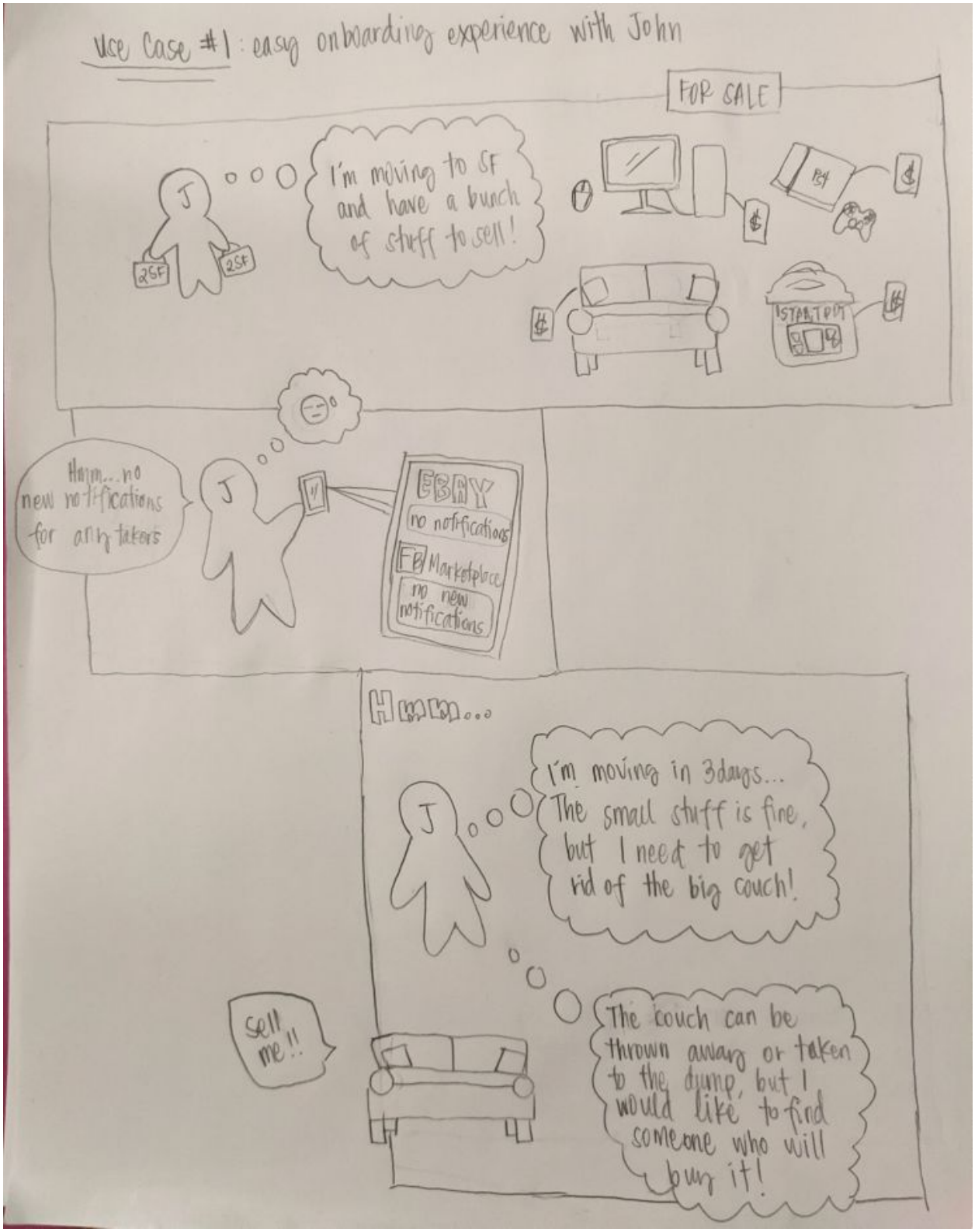

AUCTION PAGE
DROPSELL AUCTION
43 Results
Sort By
$149.00
8 watching
$19.00
Best offer
$258.00
Free shipping
$1.50
Best offer
10 bids
DROPSELL AUCTION
Seller Info
APPLE IPHONE XS
64 GB GOLD
WORKS GREAT
Condition : Used
Time Left : 23h 20m
Starting bid : US$ 189.00
Current bid : US$ 202.50
Total 50 bids
Hover to Zoom
27
Place bid
Enter Amount
Add to WishList

Buyer/Seller Chat

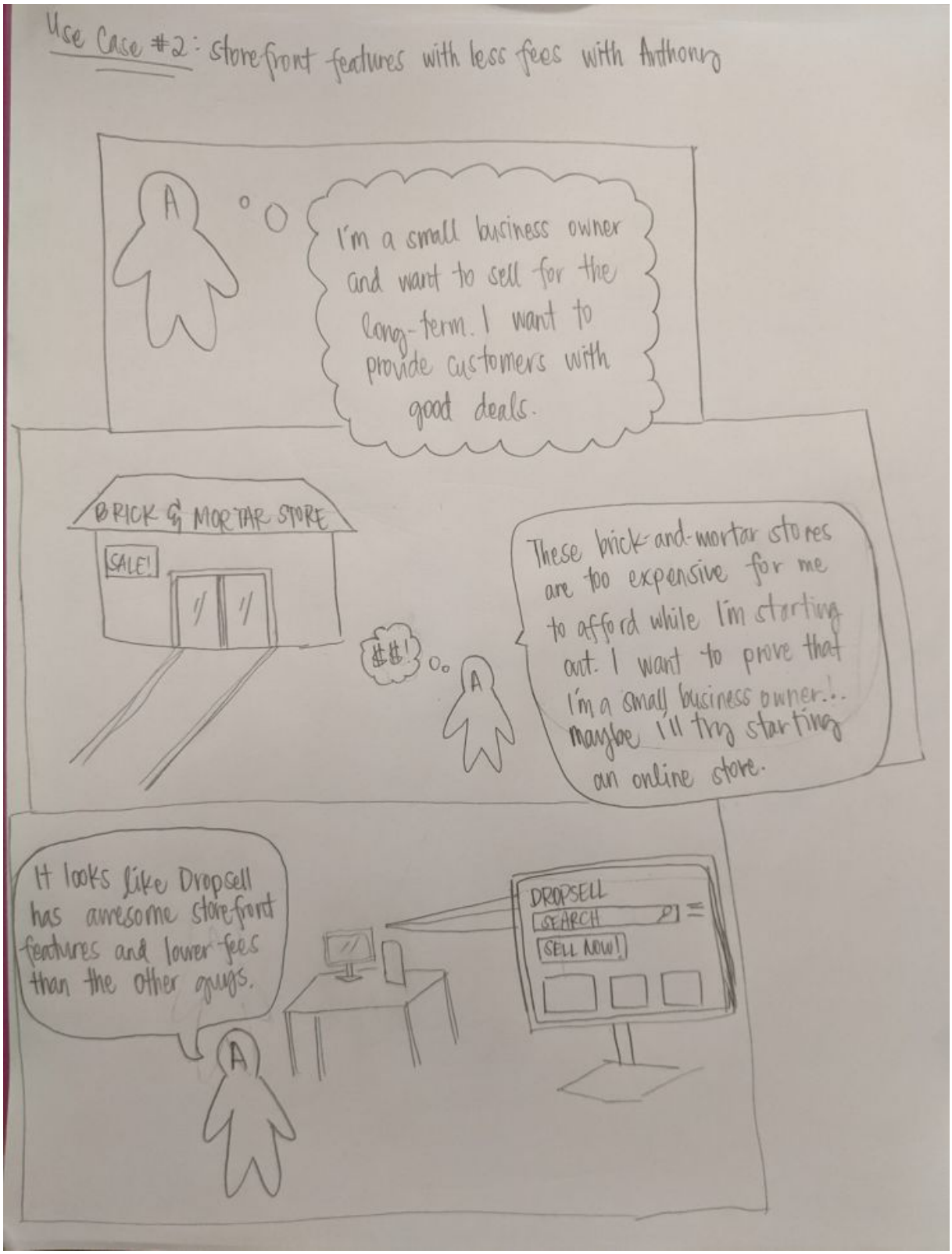


Storyboards

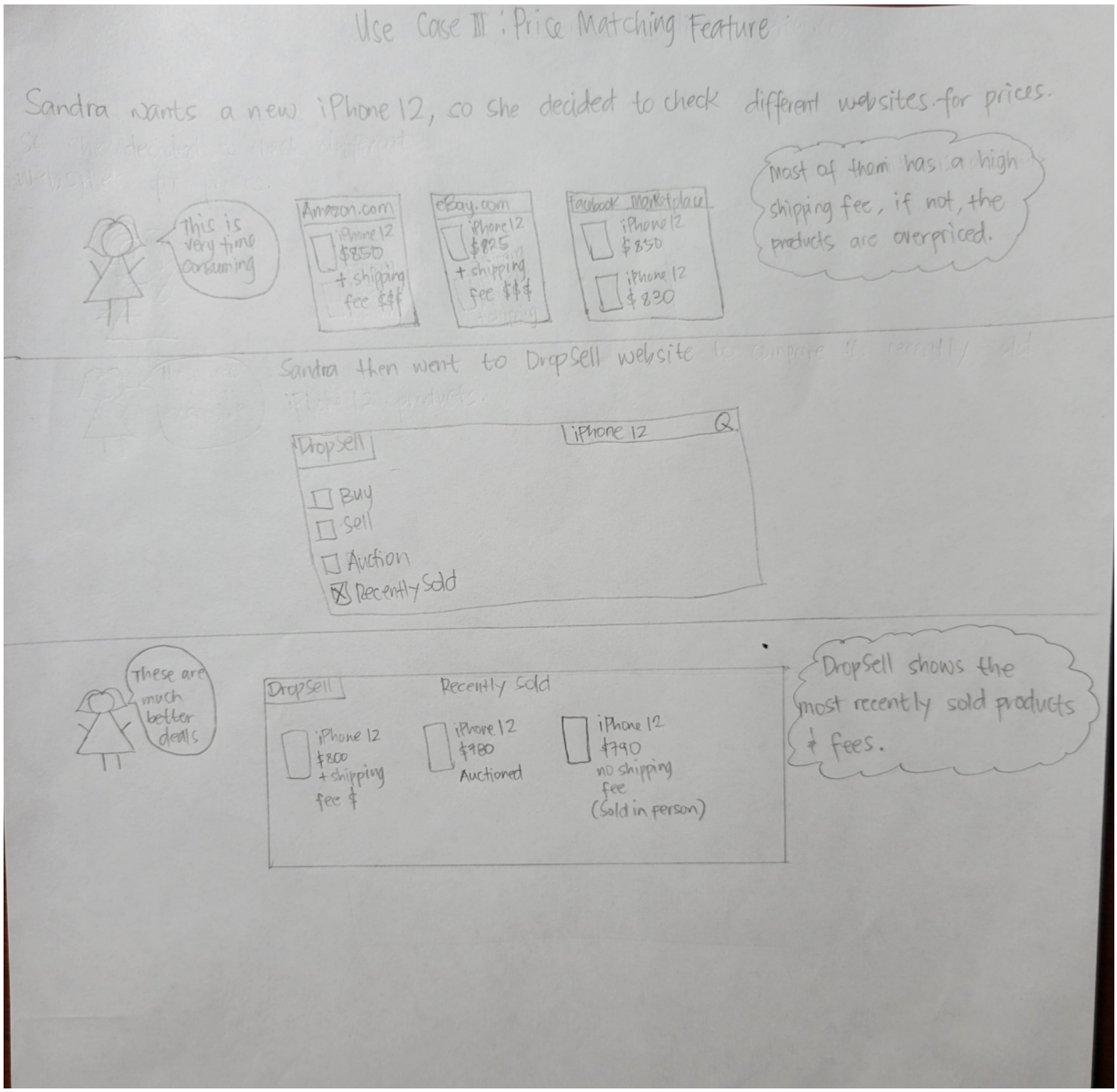
Use Case #1



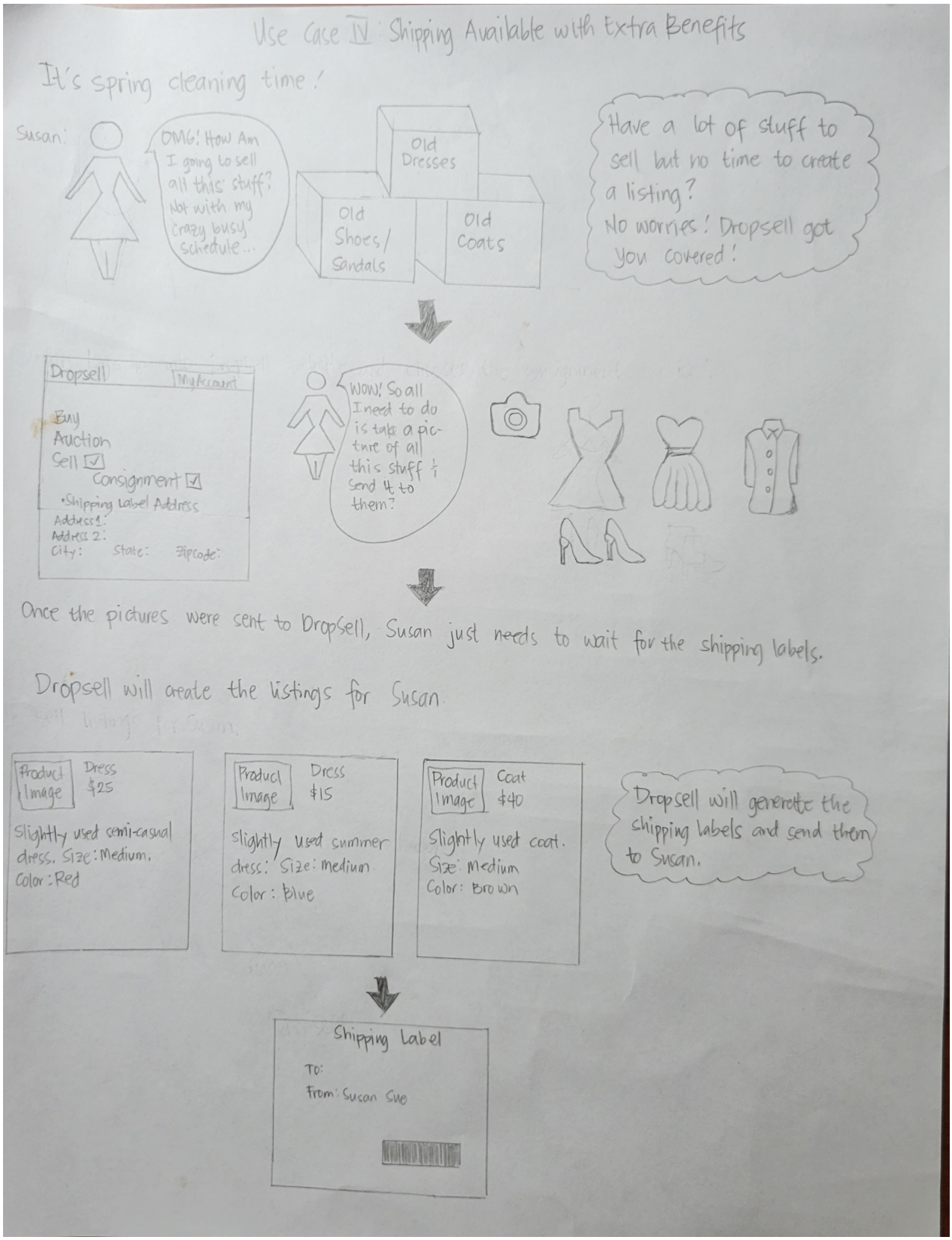
Use case #2


Use Case #2: storefront features with less fees with Anthony
A
I'm a small business owner and want to sell for the long-term. I want to provide customers with good deals.
BRICK & MORTAR STORE
SALE!
These brick-and-mortar stores are too expensive for me to afford while I'm starting out. I want to prove that I'm a small business owner... maybe I'll try starting an online store.
#$#!
A
It looks like Dropsell has awesome storefront features and lower fees than the other guys.
DROPSELL
SEARCH
SELL NOW!
A

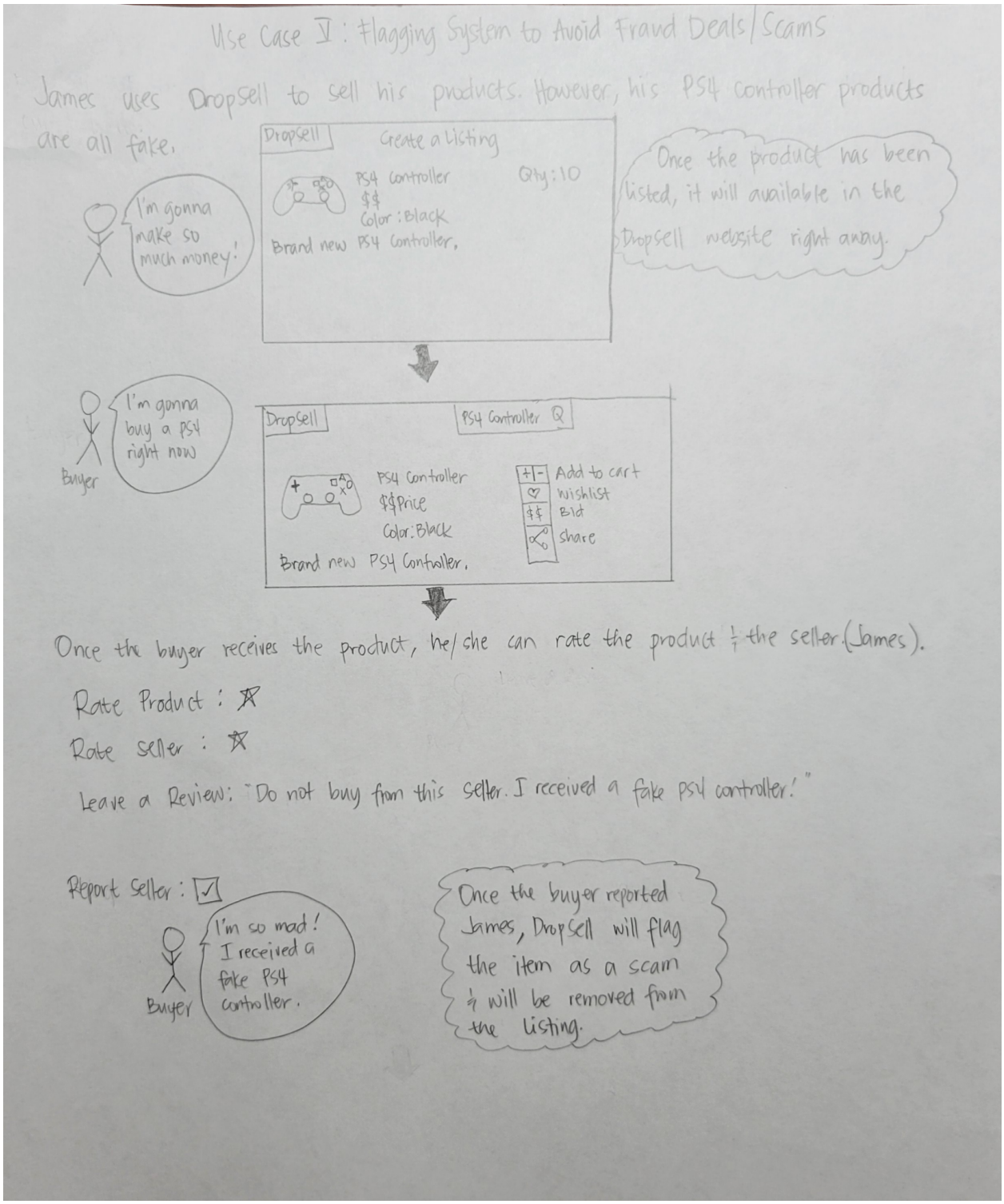
Use case #3


Use Case III : Price Matching Feature
Sandra wants a new iPhone 12, so she decided to check different websites for prices.
This is very time consuming
Amazon.com
iPhone 12
$850
+ shipping fee $$$
eBay.com
iPhone 12
$825
+ shipping fee $$$
Facebook Marketplace
iPhone 12
$850
iPhone 12
$830
Most of them has a high shipping fee, if not, the products are overpriced.
Sandra then went to DropSell website to compare the recently sold iPhone 12 products.
DropSell
iPhone 12
Buy
Sell
Auction
Recently Sold
These are much better deals
DropSell
Recently Sold
iPhone 12
$800
+ shipping fee $
iPhone 12
$980
Auctioned
iPhone 12
$790
NO shipping fee
(Sold in person)
DropSell shows the most recently sold products & fees.

Use case #4


Use Case IV: Shipping Available with Extra Benefits
It's spring cleaning time!
Susan:
OMG! How Am I going to sell all this stuff? Not with my crazy busy schedule...
Old Dresses
Old Shoes/ Sandals
Old Coats
Have a lot of stuff to sell but no time to create a listing? No worries! Dropsell got you covered!
Dropsell
My Account
Buy
Auction
Sell
Consignment
•Shipping Label Address
Address 1:
Address 2:
City: State: Zipcode:
Wow! So all I need to do is take a picture of all this stuff & send it to them?
Once the pictures were sent to DropSell, Susan just needs to wait for the shipping labels.
Dropsell will create the listings for Susan.
Product Image
Dress $25
Slightly used semi-casual dress. Size: Medium. Color: Red
Product Image
Dress $15
Slightly used summer dress: Size: Medium Color: Blue
Product Image
Coat $40
Slightly used coat. Size: Medium Color: Brown
Dropsell will generate the shipping labels and send them to Susan.
Shipping Label
To:
From: Susan Sue

Use case #5


Use Case V: Flagging System to Avoid Fraud Deals/Scams
James uses Dropsell to sell his products. However, his PS4 controller products are all fake.
I'm gonna make so much money!
Dropsell
Create a Listing
PS4 Controller
Qty: 10
$$
Color: Black
Brand new PS4 Controller.
Once the product has been listed, it will available in the Dropsell website right away.
I'm gonna buy a PS4 right now
Buyer
Dropsell
PS4 Controller Q
PS4 Controller
$$Price
Color: Black
Brand new PS4 Controller.
Add to cart
wishlist
Bid
Share
Once the buyer receives the product, he/she can rate the product & the seller (James).
Rate Product : ☆
Rate seller : ☆
Leave a Review: "Do not buy from this seller. I received a fake PS4 controller!"
Report Seller : ☑
I'm so mad! I received a fake PS4 controller.
Buyer
Once the buyer reported James, Dropsell will flag the item as a scam & will be removed from the listing.

4. High Level Database Architecture and Organization

Business Rules

1. One registered user can create zero, one, or many products. A registered user shall upload at least one image, many images are optional.
2. One registered user can have zero, one, or many seller ratings. Registered users can optionally keep track of the rating data to display in seller analytics.
3. One registered user can have one session assigned to them. User sessions can optionally store session data related to the user, in addition to mandatory session id and expiration values.
4. One registered user can have one shopping cart. It is optional for registered users to add products into their shopping cart, it is not necessary for records to exist in a user's shopping cart before a user is registered.
5. Many registered users can create zero, one, or many product comments. Registered users can optionally edit or delete their comment data.
6. Many registered users can create zero, one, or many conversations. It is optional for registered users to have conversations with other registered users, they should still have full functionality of the website.

Entities, attributes, relationships, domains

1. Registered user
    a. Attributes
        i. user_id
        ii. username
        iii. email
        iv. password
        v. first_name
        vi. last_name
        vii. phone
        viii. street
        ix. city
        x. state
        xi. zip
        xii. is_active
        xiii. created_at
        xiv. is_buyer
        xv. is_seller
    b. Relationships (Other tables)
        i. sessions
        ii. shopping_cart
        iii. conversations
        iv. messages
        v. meetups
        vi. seller_ratings
        vii. products
2. Sessions
    a. Attributes
        i. session_id
        ii. session_expires
        iii. session_data

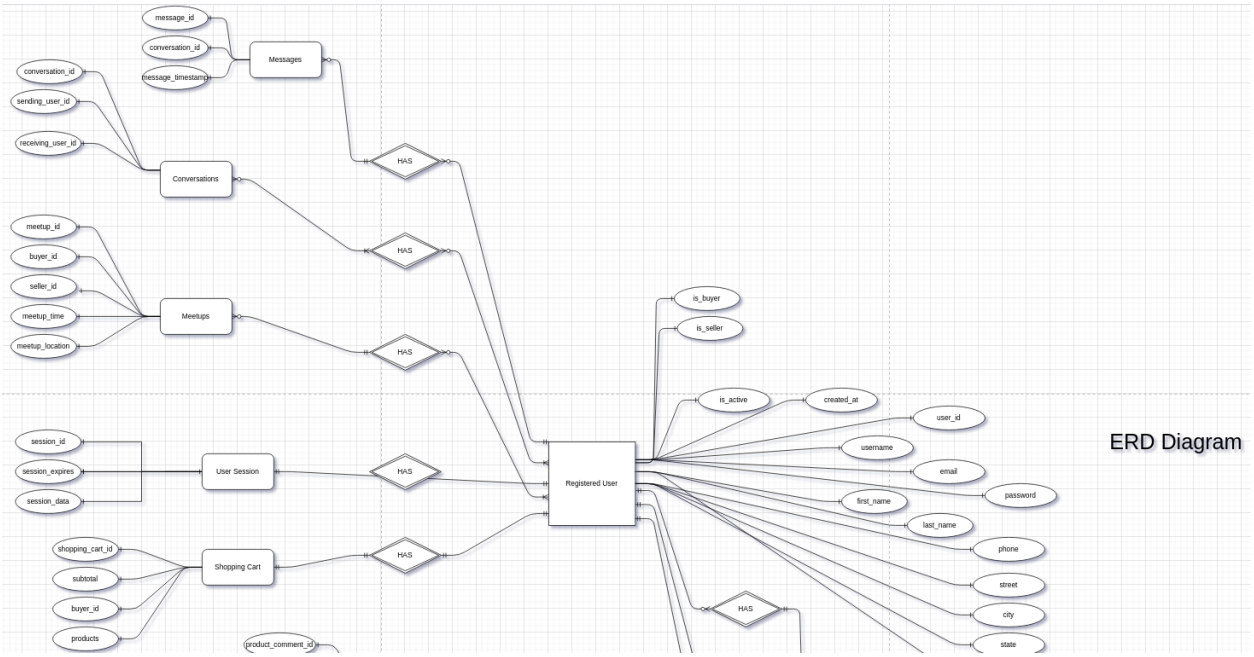
    b. Relationships
        i. users
3. Conversations
    a. Attributes
        i. conversation_id
        ii. sending_user_id
        iii. receiving_user_id
    b. Relationships
        i. users
        ii. messages
4. Messages
    a. Attributes
        i. message_id
        ii. conversation_id
        iii. message_timestamp
    b. Relationships
        i. Conversations
5. Meetups
    a. Attributes
        i. meetup_id
        ii. buyer_id
        iii. seller_id
        iv. meetup_time
        v. meetup_location
    b. Relationships
        i. users
6. Shopping cart
    a. Attributes
        i. shopping_cart_id
        ii. buyer_id
        iii. subtotal
    b. Relationships
        i. user
7. Marketplace Products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. title
        iv. description
        v. price
        vi. images
        vii. category
    b. Relationships
        i. users
        ii. product_comments
        iii. product_ratings
        iv. product_refunds
        v. top_purchased_products
        vi. daily_deal_products
        vii. shipping_products
8. Product comments

    a. Attributes
        i. product_comment_id
        ii. creator_id
        iii. product_id
        iv. comment_timestamp
        v. comment
    b. Relationships:
        i. products
        ii. users
9. Product ratings
    a. Attributes
        i. product_rating_id
        ii. product_id
        iii. creator_id
        iv. product_rating
    b. Relationships
        i. products
        ii. users
10. Product refunds
    a. Attributes
        i. product_refund_id
        ii. product_id
        iii. buyer_id
        iv. seller_id
        v. refund_amount
    b. Relationships
        i. products
        ii. users
11. Auction products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. starting_bid
        iv. auction_duration
    b. Relationships
        i. products
        ii. users
12. Top-purchased products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. total_purchased
        iv. added_at
    b. Relationships
        i. products
        ii. users
13. Daily deal products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. deal_duration

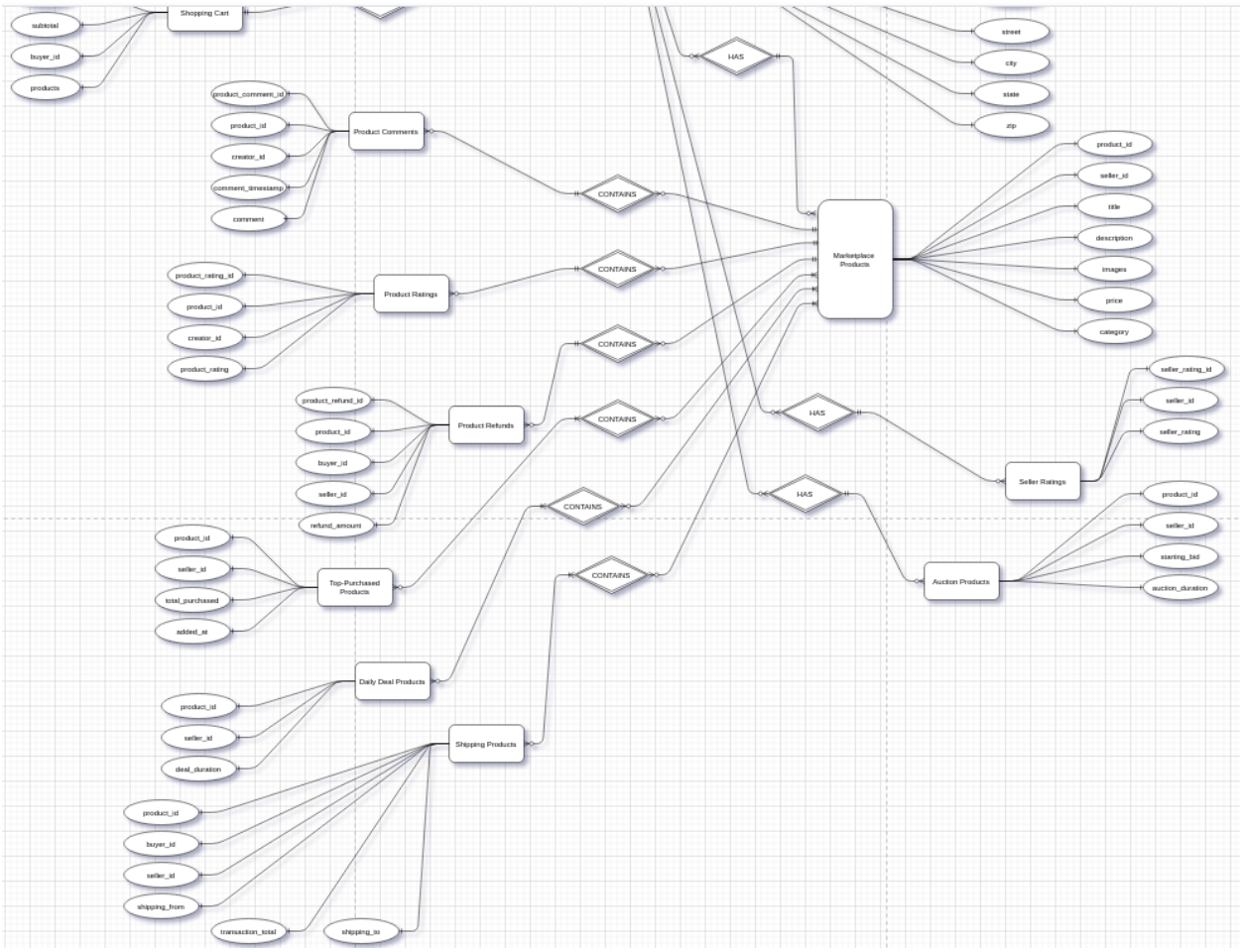
   b. Relationships
      i. products
      ii. users
14. Shipping products
   a. Attributes
      i. product_id
      ii. buyer_id
      iii. seller_id
      iv. shipping_from
      v. shipping_to
      vi. transaction_total
   b. Relationships
      i. products
      ii. users

Entity Relationship Diagram

1. https://drive.google.com/file/d/1OmkYbqwzamnWdM2J0xKMSGNLJ2dQBWrc/view?usp=sharing (top of document)

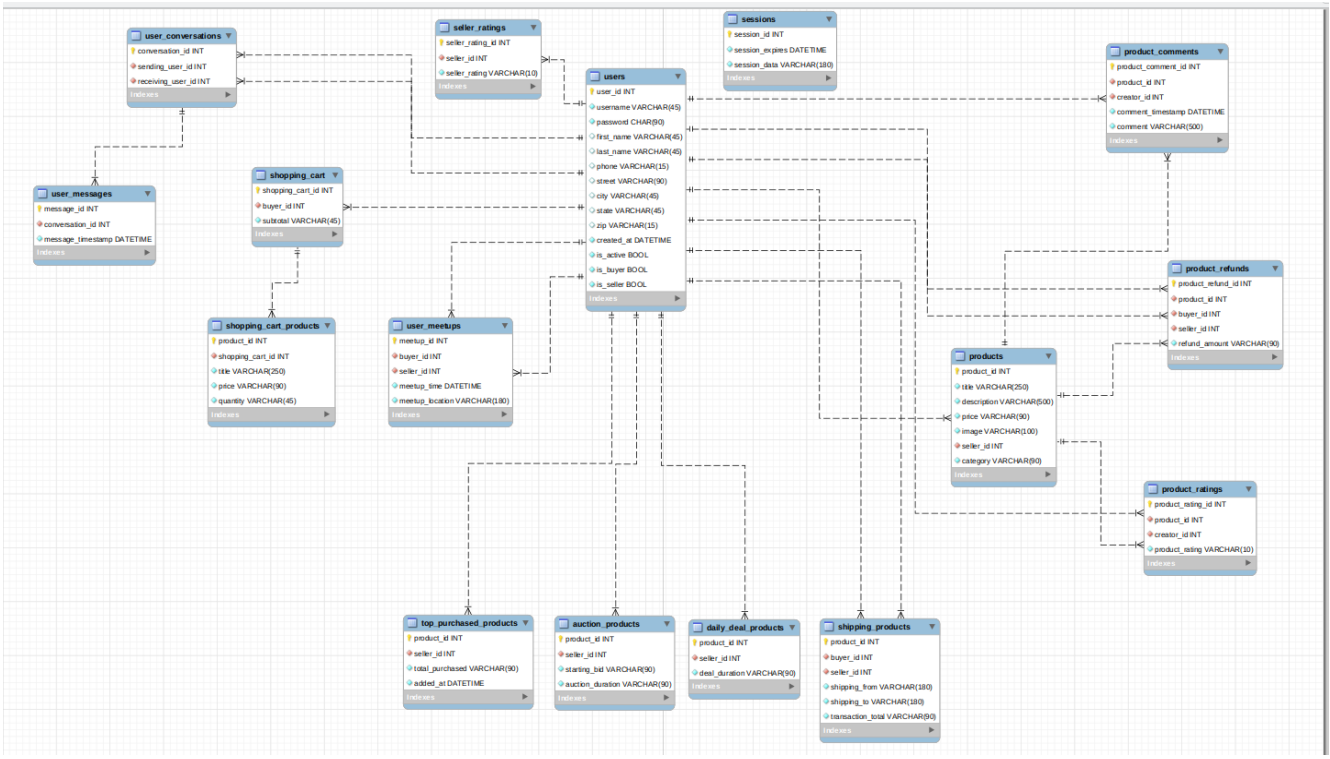


Entity Relationship Diagram



Database Model

1. https://github.com/sfsu-joseo/csc648-848-sw-engineering-sum21-T03/blob/development/application/server/db/mysql_ja_model.mwb



DBMS Decision

1. We will be using MySQL Workbench as our DBMS, since the software comes out of the box with features such as writing custom SQL statements, and creating/managing models and schemas. MySQL Workbench is also an easy-to-use interface which makes creating new tables, setting public/foreign key relationships, and creating new columns simple to explain to other team members.

Media Storage

1. Images and video/audio files will be stored in our project's file system. We will only be storing the image and video/audio file names in our database in order to save space. Additionally, we have decided to store these media files in our project's file system because we can keep them in one consistent location. As a result, this makes lookup times and references to these media files easy and efficient. Since we are storing only the media file names in the database, we can simply prepend the proper path behind the image when we have to display these media files to the client. A good example of prepending the proper path would be with an <img/> tag: src={`/uploads/${product.image}`}. When we load the src attribute, we can add /uploads to the front of the file name and we're done.

Search/filter architecture and implementation

1. Our search algorithm involves the use of a URLSearchParams object which is created when the user clicks the 'search' button of the search bar. We have implemented a filterProducts(products, query) function, which takes the current Array of products and the

search query parameter to filter with. filterProducts() then returns an updated Array with the products corresponding to the specified search parameter. By default, our Home page will load all of the products saved in the database. This is done with a ‘SELECT * FROM products’ SQL query.

2. In order to implement the category filtering for our search bar, we have implemented an initial array of categories to choose from: Clothes, Shoes, and Electronics. When a user opens the category dropdown menu, they can choose from these options to filter products. When they click one of the category options, an axios request is sent to our Node API which takes the category they clicked as a query parameter. On the backend, the ‘/api/product-categories’ route will fetch all products from the database which correspond to the category the user clicked on. This is done with a ‘SELECT * FROM products WHERE category = ?” statement, where ? is filled in with the specified category.
3. Products will be created dynamically inside our ProductCreationForm.js component. With this component, users can specify the title, description, price, category, and image of their new product. After a user creates a new product, the product will be displayed on the Home page along with any other products.

5. High Level APIs and Main Algorithms

a. API's
   i. Creation of accounts(adding to DB)
      1. The user inputs all of their new account information, such as: username, email, password, and confirm password.
      2. Once the user passes all of the client-side validation, they can then register their account with the credentials they have provided.
      3. When the user clicks the register button, this will trigger a post request to our Node API server. The user's credentials will be sent in this post request using the request body object.
      4. After the post request is sent to the server, there is server-side validation performed, which checks that all of the user's credentials are valid.
      5. After the user's credentials are valid, this data will be added into an async/await call or a Promise object which will trigger the SQL statement which adds the new user into the database.
      6. If there were no errors when creating the new user, then the user should be directed to the login screen to login.
   ii. Logging into accounts(accessing DB/verification)
      1. Input username/password, after clicking login then send post request to node api server, check to see if the user exists in the database, if it doesn't return error message to user, if it does, create a new session for the user, and then redirect them to their Home page
   iii. Similar products
      1. When a user is browsing the application, store a history of the last 10 products they viewed. Use this information to display suggestions to the user.
   iv. Buyers reviews, ratings.
      1. When users create reviews for products that review is stored in the database and linked to the appropriate item
      2. Ratings are based on a 5 star system.
         a. As a registered user rates another registered user they leave a review that gets stored in the database and linked to the registered user.
         b. Registered users can also leave a rating between 1-5 for that registered user that will be stored in the database linked to the appropriate user.
         c. All reviews and ratings will be attached to profiles and when any user loads a profile this information will be retrieved from the database and displayed.
   v. Shopping cart
      1. When a user adds an item to their shopping cart that item is reserved for them for a period of 10 minutes(can be adjusted or tied to a user's session).
         a. OPTION A, when placed in a shopping cart that item is reserved to that user for a period of time so that another user can't accidently buy the product before them if they check out faster.
         b. OPTION B, we inform the user how many users concurrently have that item in their shopping cart and establish(and importantly inform the user) that their item is not secured until they start the actual check out/payment process. This means another user can accidently steal another usersitem if they check out first.
   vi. Messaging.

1. Array of objects, with each object containing the ID of the message poster, ID of the message itself, and the time the message was sent.

vii. Tag system/price checking
1. When an item is created the seller will enter all appropriate tags to the item in a fashion similar to hashtags. For example a PS5 might have the tags; #Electronics,#Game,#Sony(# used for reference).
2. Using tags we can program different algorithms such as daily deals, categories, price checking.
3. This tag will be stored with the item in the database,
4. Users must enter a minimum number of tags.

viii. Listing products and retrieving product information: title, description, image, price, brand, condition(new/used/etc), misc/notes, and importantly any tags that apply to the product.
1. Creating a new product, after checking that all required sections are filled in(title, description, image, price, brand, condition, misc/notes, and tags.), create a database entry linked to the user.
2. If the request to see a product is valid, retrieve the product information and display it in the appropriate manner(ie seller/buyer history, auction listing, etc)
3. When a product is created for sale, before submitting the item to the database/market place run the price checking algo and display suggested prices and similar product information.

ix. Buyer Receipt details for delivery or checkout (first name, last name, phone, email)
1. If user elected for peer-to-peer deliver simply store the transaction information in the users history
2. If a user opted for a consignment service then, like the peer-to-peer service, the transaction information will be stored in the user's history. Then a shipping label will be provided to the buyer.

x. Inappropriate language detection
1. Maintain a list of words/phrases that are deemed inappropriate or that violate terms and conditions.
2. When a user creates content(i.e. comments on a post, leaves review, creates a product, etc) parse the text into tokens and check against our list of inappropriate language.
3. Handle the inappropriate comment based on our terms/condition.

b. Significant non-trivial Algorithm/process

i. Price Matching against similar products(Superior service)
1. Using the Tag api to get product tags.
2. Compare the tags listed on the product with similar tagged items
    a. When comparing tags we will look at similar items that have been listed in the past(time period to be adjusted, up to 3 months).
    b. Based on the tags of the item compared to current and past tags of similar items we generate a list of information and present that to the user.
    c. This information will show simple statistics of sales related to the item.
    d. Display the picture of the item so the seller can see how relatable their item is to the suggested ones.
3. When a seller posts an item our price checker will display listings that have sold in the past, are currently listed, and products that are not selling and all associated price points of those objects.
4. This price checking allows a new or inexperienced seller to get a feel for the correct amount they should list their item for.

a. Example 1, a user wants to sell an item fast.
    i. They list the item, our price checking algorithm displays history of sells and current listings
    ii. The user sees that the item averages 100$ over the past 3 months.
    iii. The seller sees that current listings of that item range from 85-90$ and that items priced over 105$ generally stay on the marketplace for 2 weeks.
    iv. Seller values time over efficiency and lists the product for 75$ to make the fastest money.
b. Example 2, a user wants the most money and is patient,
    i. User has an item sitting around in their storage and wants to make the most money they can, and are willing to wait
    ii. They list the item, our price checking algorithm displays history of sales and current listings.
    iii. The user sees that the item averages 500$ over the past 3 months
    iv. The user sees that over the past 3 months items placed over 575 take an average of 3 weeks-1 month to sell.
    v. The user also sees that items placed over 600$ rarely sell.
    vi. The user is patient and lists the item for 580$ to get the biggest return.
c. Example 3, a user is new and has an unique item they don't know how to price.
    i. The user has never sold anything second hand before and has a high quality painting they wish to sell.
    ii. They list the item, our price checking algorithm displays history of sells and current listings
    iii. The user sees similar paintings for this artist but does not see the exact match for his painting.
    iv. Based on the information return the user can see an average cost of items of similar nature.
    v. Using this information he knows that if he lists the painting for under 200$ he will most likely be losing money and that if he lists the painting for over 700$ he may never sell it.
    vi. The user lists the painting for 400$.

ii. Ordering reviews and rates
    1. This will help to organize reviews and rates
        i. When buyers purchase their items, they are able to review and rate their items and post them on a public page.
        ii. After their review and rate items end, then it will post them to the public in alphabetical order to make sure it is easy to find and organize for reviews and rates items.

iii. Alerting if most of the items have bad reviews and rates
    1. Let the sellers know that their selling items must be satisfied to the buyers.
        i. This can have feedback from the buyers, and making sure that when the sellers sell their items, they have to improve and make better items of what they are selling.

2. If the sellers cannot resolve this issue, then let the business management know about this issue.
    i. Having bad reviews and rates can cause down business selling items.
    ii. Sellers should be very responsible to make sure that their items are good enough.

iv. Detecting inappropriate comments
    1. When the buyers put comments on selling products, this algorithm will avoid inappropriate comments as possible.
        i. This is useful because when young children visit this site, then they should NOT read any inappropriate comments.
    2. Use API Calls from the list of inappropriate words, to see if one or more words matches the comments, then the comments should not be allowed to post them in public.

v. Recommend best meetup time
    1. When the buyers and sellers decide to schedule meetup times, then recommend the best meetup times will show the best of what days and times to meet.
        i. Just in case the buyers or sellers are too lazy to set up their scheduled meetings. They can just click “Show best meeting times”.

    2. Use a graph algorithm to optimize the best meetup times between the seller and buyer by analyzing their schedule planner.

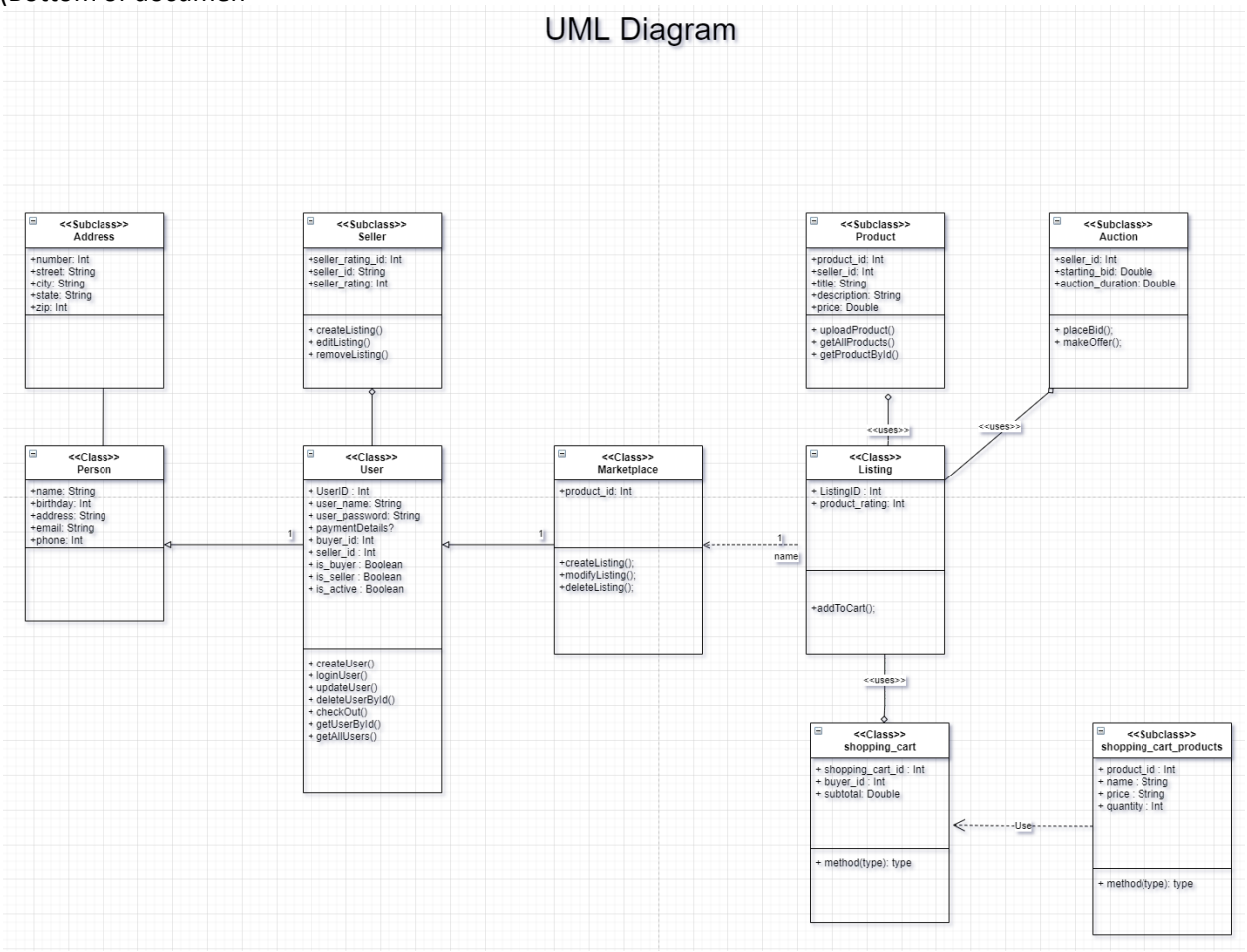
vi. Calculate the mean and standard deviation of the number of selling, buying, refund, and return products.
    1. Keeping track of a fair number of selling, buying, refund, and return products.
    2. This will help to manage business and keep track of customers and sellers actions.

vii. Alerting if most of the items have many reports and complaints from buyers.
    1. If most of the buyers report their items, then the sellers and business management let know that they should be aware to resolve this issue.

viii. Match categories for all products
    1. Making sure what kind of products belong to which categories
        i. For example, if the seller sells a pencil and the categories are school, work, home, and etc. The algorithm will figure out what pencil does relate to one of these categories. Pencil item should be in the school category.
    2. This helps to organize and search for buyers' wanted items.

6. High Level UML Diagrams

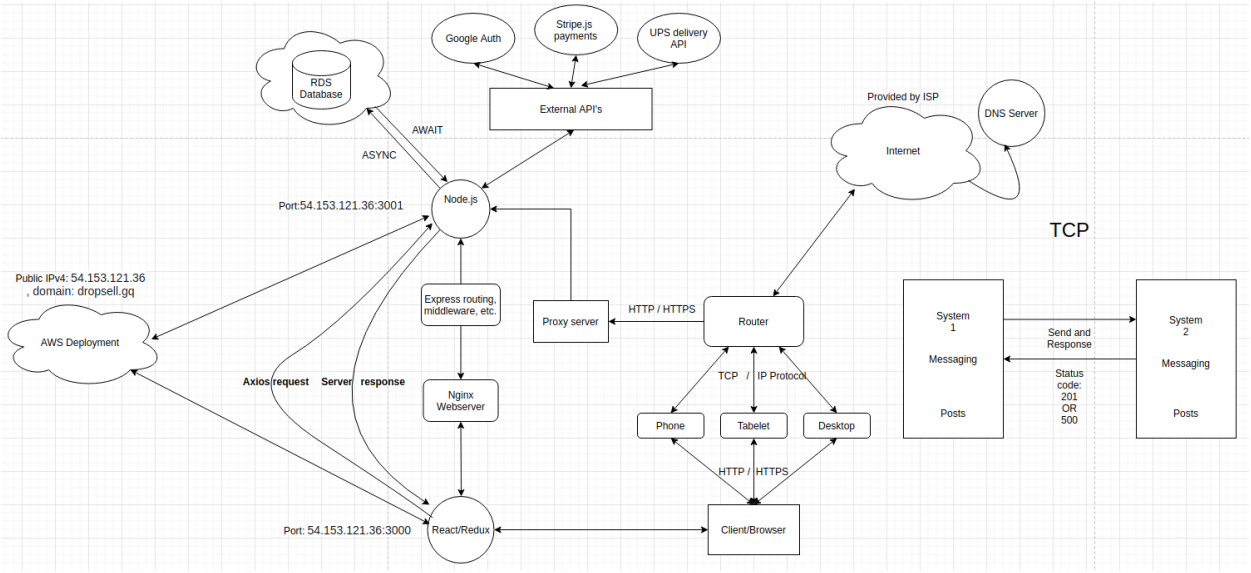
1. https://drive.google.com/file/d/1OmkYbqwzamnWdM2J0xKMSGNLJ2dQBWrc/view?usp=sharing
   (Bottom of documen



7. High Level Application Network and Deployment Diagrams

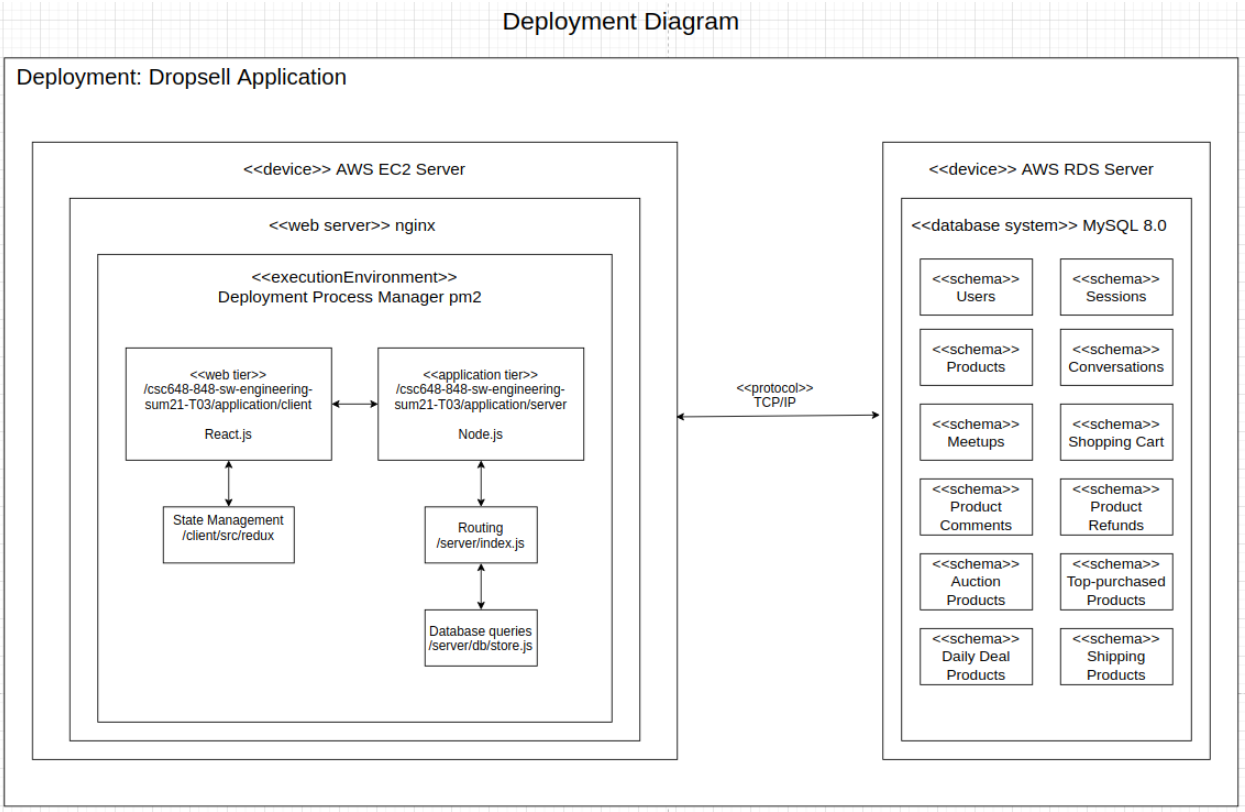
Application Network

1. https://drive.google.com/file/d/1XLbonkJqFnO7ZeOMQqstFPeZm8cr7UxX/view?usp=sharing
   (Top of documen



Deployment diagram

1. https://drive.google.com/file/d/1XLbonkJqFnO7ZeOMQqstFPeZm8cr7UxX/view?usp=sharing
   (Bottom of document)



8. Identify actual key risks for your project at this time

a. Schedule risks
    i. There are conflicting schedules when trying to set up group meetings. Our group will mitigate this by working both asynchronous and synchronous when possible.
b. Legal risks
    i. This is related to either copyright laws when saving images or copying another company's idea. We will resolve this by either making our content as unique as possible, or obtaining the appropriate copyrights and/or licensing.
c. User risks
    i. This is related to scammers who wrongfully use the application. To resolve this issue, we will add security such as captcha, email verification, two-factor authentication, and a report button which users will be able to utilize accordingly.
d. Selling items risk
    i. This unique risk comes with many businesses set up for users to sell their products. Some products can possibly be dangerous and illegal to sell. We will resolve this by labeling such products as dangerous and/or illegal.
e. Skills risk
    i. As with any group work, there will be different skill levels amongst the members. We will mitigate this by sharing information, teaching each other, and working together in a way that best benefits what each member is the most capable of doing.

9. Project Management

As a team, our group decided the best way to complete this milestone was to take a divide and conquer approach. We worked together synchronously as much as possible, asynchronously when we were not able to meet, and communicated through Discord throughout. During meetings, we discussed how we would prioritize our functional requirements and categorized them accordingly. We split up the UI mockups and storyboards initially, then split up work on the database, UML and network diagrams, storyboards and editing the milestone document. We messaged each other in Discord concerning updates. We had set meetings for any questions, and discussed backlogs (upcoming lists of tasks we want done), new features, running tasks (tasks under development, tasks that were fixed and/or upgraded, tasks ready to move to production, and tasks that were labeled as done and/or deployed.) We set up and utilized the Trello application for managing all these tasks as a group. Overall, the methods that our group has been using to complete the milestones have worked really well so far. We will continue to use Trello, have scheduled meetings to manage current and future tasks, as well as maintain daily communication on Discord

10. Detailed list of contributions

Student Name	Contributions
Mitchel Baker	Mitchel started off M2 working on the UI mockups for our user checkout experience, which included: receipt info, summary, checkout, and final invoice pages. He collaborated with Michael to complete the high level database architecture requirements, ranging from our business rules, ERD diagram, database model, DBMS of choice, media storage decision, and the search/filter architecture. Mitchel also worked with Kenneth in order to research/design our application network diagram, while also creating the deployment diagram, list of contributions, search/filter implementation, and the vertical SW prototype. Lastly, Mitchel wrote up the data definitions and filled in necessary details.
Charmaine Eusebio	Charmaine created a few of the most important UI/UX web pages for our site. She wrote up from scratch the seller analytics (key feature), the create product page, list of auctions, products for sale, starred/watching product and starred/watching list. She also played a pivotal role with the storyboards which reflected our use cases 1 and 2. Lastly, Charmaine identified many of the key risks which she found sufficient solutions for, while also discussing our project management style.
Kenneth N Chuson	Kenneth did a tremendous job with the amount of detail he put into the profile and user settings pages. You can see the amount of work he put into them as a result of his mockup's quality. Kenneth also played a huge role with the write up of our high level API's and main algorithms. He has brought up numerous ideas involving data analytics and machine learning which will really make our application stand out. Lastly,

	Kenneth conducted the majority of the research for creating our application network diagram.
Krina Bharatbhai Patel	Krina jumpstarted our progress on Milestone 2 by creating our initial set of data definitions. She laid the groundwork for the future additions added into our data definitions section. Krina also demonstrated her creativity with the login, register, auction, and chat mockups. Her mockups display all of the key features which we want to implement for our application, while showing off a solid understanding of UI/UX fundamentals. Krina was also responsible for prioritizing our functional requirements. Despite not being an easy task, she took to it quickly and showed that she can deliver quality work with efficiency. Lastly, Krina was also a big part of creating the vertical prototype by implementing a couple of the main UI components for our home page.
Michael Schroeder	Michael was assigned the creation of our initial home and menu page mockups. He took to these tasks with diligence and demonstrated his ability as a quick learner. Michael also collaborated with Mitchel on the ERD diagram portion of the high level database architecture requirements, while also implementing from scratch the UML diagram for our project. Despite not having prior experience writing UML diagrams, he demonstrated an eagerness to learn new skills which is a rare skill to find in people.
Rowena Elaine Echevarria	Rowena was in charge of writing up the about page for our UI mockups. She also did a tremendous job with the creation of her storyboards for use cases 3, 4, 5. She incorporated a ton of detail into her storyboards, which demonstrates her attention to detail, especially in regards to UI/UX. As a result of the detail put into her storyboards, our Milestone 2 gives readers the opportunity to see first-hand examples of how Dropsell

	solves problems for users who are looking to either buy or sell products online.
Jamie Dominic Walker	Jamie was in charge of the high level API and main algorithms section. He demonstrated an eagerness to learn more about API's by researching and sharing videos/articles about the topic. Jamie is someone who knows how to apply his critical thinking skills to real-world applications. When it comes to explaining his thoughts and ideas, Jamie takes an implementation-based approach to explaining our main algorithms which was extremely helpful. He explored various edge cases related to our API implementations, while also bringing up thought-provoking analysis during our meeting discussions.

M3

SW Engineering CSC648-848 Summer 2021
dropsell.gq, Jose's Angels

Team 03

Name	Email	Roles
Mitchel Baker	mbaker3@mail.sfsu.edu	Team lead
Krina Bharatbhai Patel	kpatel11@sfsu.edu	Frontend lead
Charmaine Eusebio	ceusebio1@mail.sfsu.edu	Frontend engineer
Rowena Elaine Echevarria	rechevarria@mail.sfsu.edu	Frontend engineer
Michael Schroeder	mschroeder@mail.sfsu.edu	Backend lead, Github master
Kenneth N Chuson	kchuson@mail.sfsu.edu	Backend engineer
Jamie Dominic Walker	jwalker5@mail.sfsu.edu	Backend engineer

Milestone 3
July 22, 2021

History Table

Version	Date	Notes
M3V2	08/02/2021	
M3V1	07/22/2021	
M2V2	07/20/2021	
M2V1	07/08/2021	
M1V2	07/02/2021	
M1V1	06/22/2021	

1. M3 Data Definitions

1. Unregistered User: A user who can visit the website, explore the products displayed in the marketplace, and checkout any public facing pages. Unregistered users need to register to use all the features of the website related to viewing auctions, commenting and rating products, messaging buyers/sellers, and purchasing products.
    a. A general user shall be able to create an account.
    b. A general user shall choose the option of creating an account as buyer/seller/both.

2. Registered User: A registered user is able to access the website with all features. Registered users need to login to the website for buying or selling products.
    a. User login
        i. username; type VARCHAR(45), NN
            Must enter username to login
        ii. password; CHAR(90), NN
            Must enter a valid password, encrypted with Bcrypt library

    b. User account details
        i. user_id; type INT, PK, NN, AI
        ii. email; type VARCHAR(90), NN
        iii. first_name; type VARCHAR(45), null
        iv. last_name; type VARCHAR(45), null
        v. phone; type VARCHAR(15), null
        vi. street; type VARCHAR(90), null
        vii. city; type VARCHAR(45), null
        viii. state; type VARCHAR(45), null
        ix. zip; type VARCHAR(15), null
        x. created_at; type DATETIME, NN
        xi. is_active; BOOL, NN
        xii. Payment details
            Attributes
                bCC number
                Expiration date
                3 digit code
                Zip
            Payment details should be handled by Stripe.js. We may store payment details in the database for multiple payment options in the future.
        xiii. User is a buyer? Seller?
            is_buyer; type BOOL, NN
            is_seller; type BOOL, NN

    c. User conversations
        i. conversation_id; type INT, PK, NN, AI
        ii. sending_user_id (FK to sending user); type INT, NN
        iii. receiving_user_id (FK to receiving user); type INT, NN
        iv. Messages
            message_id; type INT, PK, NN, AI
            conversation_id (FK to conversation id); type INT, NN
            message_timestamp; type DATETIME, NN

   d. User meetups
      i. meetup_id; type INT, PK, NN, AI
      ii. buyer_id (FK to buyer id); type INT, NN
      iii. seller_id (FK to seller id); type INT, NN
      iv. meetup_time; type DATETIME, NN
      v. meetup_location; type VARCHAR(180), NN

3. User session
   a. Users should have an active session created in order to keep track of the date and time they logged in. This information will be used to keep track of the length of the user's session.
   b. When the user logs in, their session data should be updated in the database.
   c. If the user's session has expired, then the user should be required to log in again.
   d. If, for example, the user's session is still active and they decide to refresh the page, then the user should stay logged in.
   e. Attributes
      i. session_id; type INT, PK, NN
      ii. session_expires; type DATETIME, NN
      iii. session_data; type VARCHAR(180), NN

4. Buyers: can browse products and buy them.
   a. shopping_cart
      i. shopping_cart_id; type INT, PK, NN, AI
      ii. buyer_id (FK to buyer id); type INT, NN
      iii. subtotal; type VARCHAR(45), NN
      iv. shopping_cart_products
         product_id; type INT, PK, NN, AI
         shopping_cart_id (FK to shopping cart id); type INT, NN
         title; type VARCHAR(250), NN
         price; type VARCHAR(90), NN
         quantity; type VARCHAR(45), NN

5. Sellers: can upload product information and sell them.
   a. Seller Ratings
      i. seller_rating_id; type INT, PK, NN, AI
      ii. seller_id; type INT, NN
      iii. seller_rating (5-star rating system, from 1-5); type VARCHAR(10), NN

   b. Products: The items which are uploaded by sellers, and purchased by buyers.
      i. product_id; type INT, PK, NN, AI
      ii. seller_id; type INT, NN
      iii. title; type VARCHAR(250), NN
         At Least 5 word Title or name of the product
      iv. description; type VARCHAR(500), NN
         At Least 10 word Description of the product
      v. image; type VARCHAR(100), NN
         At least 3 images of size (600x600 or 2x2) for product visibility
      vi. price; type VARCHAR(90), NN
         The price for product (For sale and auction only)
      vii. category; type VARCHAR(90), NN

c. Product Comments
    i. product_comment_id; type INT, PK, NN, AI
    ii. product_id (FK to product id); type INT, NN
    iii. creator_id (FK to user id); type INT, NN
    iv. comment_timestamp; type DATETIME, NN
    v. comment; type VARCHAR(500); NN

d. Product Ratings
    i. product_rating_id; type INT, PK, NN, AI
    ii. product_id (FK to product id); type INT, NN
    iii. creator_id (FK to user id); type INT, NN
    iv. product_rating (5-star rating system, from 1-5); type VARCHAR(10), NN

e. Product Refunds
    i. product_refund_id; type INT, PK, NN, AI
    ii. product_id (FK to product id); type INT, NN
    iii. buyer_id (FK to buyer id); type INT, NN
    iv. seller_id (FK to seller id); type INT, NN
    v. refund_amount; type VARCHAR(90), NN

6. Auction Products
    a. product_id; type INT, PK, NN, AI
    b. seller_id (FK to seller id); type INT, NN
    c. starting_bid; type VARCHAR(90), NN
    d. auction_duration: type VARCHAR(90), NN

7. Top-Purchased Products
    a. product_id; type INT, PK, NN, AI
    b. seller_id (FK to seller id); type INT, NN
    c. total_purchased; VARCHAR(90), NN
    d. added_at; type DATETIME, NN

8. Daily Deal Products
    a. product_id; type INT, PK, NN, AI
    b. seller_id (FK to seller id); type INT, NN
    c. deal_duration: type VARCHAR(90), NN

9. Shipping Products
    a. product_id; type INT, PK, NN, AI
    b. buyer_id (FK to buyer id); type INT, NN
    c. seller_id (FK to seller id); type INT, NN
    d. shipping_from; type VARCHAR(180), NN
    e. shipping_to; type VARCHAR(180), NN
    f. transaction_total; type VARCHAR(90), NN

10. Redux related data definitions
    a. Login
        i. Actions
            setUsername(username)
                Action type: ‘USER_SET_USERNAME’
                Datatype: String

setPassword(password)
Action type: ‘USER_SET_PASSWORD’
Datatype: String
loginUser()
userData(username, password)
redirectUserAfterLogin(loggedIn)
Action type: ‘USER_IS_LOGGEDIN’
Datatype: String

ii. Reducer

INITIAL_LOGIN_STATE
Username; datatype String
Password; datatype String
loggedIn; datatype Boolean

b. Register

i. Actions

setUsername(username)
Action type: ‘USER_SET_USERNAME’
Datatype: String
setPassword(password)
Action type: ‘USER_SET_PASSWORD’
Datatype: String
setConfirmPassword(confirmPassword)
Action type: ‘USER_SET_CONFIRM_PASSWORD’
Datatype: String
createUser()
userData(username, password, confirmPassword)
redirectUser(registered)
Action type: ‘USER_IS_REGISTERED’
Datatype: Boolean

ii. Reducer

INITIAL_REGISTER_STATE
Username; datatype String
Password; datatype String
confirmPassword; datatype String
Registered; datatype Boolean

c. Products

i. Actions

setTitle(title)
Action type: ‘PRODUCT_SET_TITLE’
Datatype: String
setDescription(description)
Action type: ‘PRODUCT_SET_DESCRIPTION’
Datatype: String
setPrice(price)
Action type: ‘PRODUCT_SET_PRICE’
Datatype: String
setImage(image)
Action type: ‘PRODUCT_SET_IMAGE’
Datatype: String

setSuccess(isSuccess)
Action type: ‘PRODUCT_SET_SUCCESS’
Datatype: Boolean
setCategory(category)
Action type: ‘PRODUCT_SET_CATEGORY’
Datatype: String
setCategories(categories)
Action type: ‘SET_CATEGORIES’
Datatype: Boolean
changeDropdownText(text)
Action type: ‘CHANGE_DROPDOWN_TEXT’
Datatype: String
createProduct()
formData(title, description, price, category, file)
getProducts(products)
Action type: ‘GET_PRODUCTS’
Datatype: Array

ii. Reducer
INITIAL_PRODUCT_STATE
Title; datatype String
Description; datatype String
Price; datatype String
Category; datatype String
File: datatype String
filePreview; datatype null/URL object
isSuccess; datatype, null/boolean
Products; datatype Array
Categories; datatype Boolean
dropdownText; datatype String

d. Seller Settings
i. Actions
1. updateFirstName(firstName)
a. Action type: ‘USER_UPDATE_FIRSTNAME’’
b. Datatype: String
updateLastName(lastName)
a. Action type: ‘USER_UPDATE_LASTNAME’
b. Datatype: String
updateBirthday(birthday)
a. Action type: ‘USER_UPDATE_BIRTHDAY’
b. Datatype: Date
4. updateEmail(email)
a. Action type: ‘USER_UPDATE_EMAIL’
b. Datatype: String
5. updatePhone(phone)
a. Action type: ‘USER_UPDATE_PHONE’
b. Datatype: Character
6. updateUserName(userName)
a. Action type: ‘USER_UPDATE_USERNAME’’
b. Datatype: String
7. updatePassword(password)
.

a. Action type: ‘USER_UPDATE_PASSWORD’’
b. Datatype: String

8. updateCardNumber(cardNumber)
    a. Action type: ‘USER_UPDATE_CARDNUMBER’
    b. Datatype: Integer
9. updateExpirationDate(cardExpiration)
    a. Action type: ‘USER_UPDATE_CARDEXPIRATION’
    b. Datatype: Date
10. updateCVV(cardCVV)
    a. Action type: ‘USER_UPDATE_CARDCVV’
    b. Datatype: Integer
11. updatePostalCode(postalCode)
    a. Action type: ‘USER_UPDATE_POSTALCODE’
    b. Datatype: Integer
12. updateBioDescription(bioDescription)
    a. Action type: ‘USER_UPDATE_BIODESCRIPTION’
    b. Datatype: String
13. updateLocation(location)
    a. Action type: ‘USER_UPDATE_LOCATION’
    b. Datatype: String
14. updateSocialMedia(socialMedia)
    a. Action type: ‘USER_UPDATE_LOCATION’
    b. Datatype: String
15. updateNoteSchedule(noteSchedule)
    a. Action type: ‘USER_UPDATE_NOTESCHEDULE’
    b. Datatype: Array

ii. Reducer

1. INITIAL_SELLER_SETTINGS_STATE
    firstName; datatype String
    lastName; datatype String
    birthday; datatype Date
    email; datatype Integer
    phone; datatype Character
    userName; datatype String
    password; datatype String
    cardNumber; datatype Integer
    cardExpiration; datatype Date
    updateCVV; datatype Integer
    updatePostalCode; datatype Integer
    bioDescription; datatype String
    location; datatype String
    socialMedia: datatype String
    noteSchedule; datatype Array

e. Buyer Settings

i. Actions

1. updateFirstName(firstName)
    a. Action type: ‘USER_UPDATE_FIRSTNAME’’
    b. Datatype: String
2. updateLastName(lastName)

a. Action type: ‘USER_UPDATE_LASTNAME’
b. Datatype: String

3. updateBirthday(birthday)
    a. Action type: ‘USER_UPDATE_BIRTHDAY’
    b. Datatype: Date
4. updateEmail(email)
    a. Action type: ‘USER_UPDATE_EMAIL’
    b. Datatype: String
5. updatePhone(phone)
    a. Action type: ‘USER_UPDATE_PHONE’
    b. Datatype: Character
6. updateUserName(userName)
    a. Action type: ‘USER_UPDATE_USERNAME’’
    b. Datatype: String
7. updatePassword(password)
    a. Action type: ‘USER_UPDATE_PASSWORD’’
    b. Datatype: String
8. updateCardNumber(cardNumber)
    a. Action type: ‘USER_UPDATE_CARDNUMBER’
    b. Datatype: Integer
9. updateExpirationDate(cardExpiration)
    a. Action type: ‘USER_UPDATE_CARDEXPIRATION’
    b. Datatype: Date
10. updateCVV(cardCVV)
    a. Action type: ‘USER_UPDATE_CARDCVV’
    b. Datatype: Integer
11. updatePostalCode(postalCode)
    a. Action type: ‘USER_UPDATE_POSTALCODE’
    b. Datatype: Integer
12. updateBioDescription(bioDescription)
    a. Action type: ‘USER_UPDATE_BIODESCRIPTION’
    b. Datatype: String
13. updateRateStars(RateStars)
    a. Action type: ‘USER_UPDATE_RATESTARS’
    b. Datatype: Integer
14. updateReview(review)
    a. Action type: ‘USER_UPDATE_REVIEW’
    b. Datatype: String
15. updateSocialMedia(socialMedia)
    a. Action type: ‘USER_UPDATE_SOCIALMEDIA’
    b. Datatype: String
16. updateShowBuys(showBuys)
    a. Action type: ‘USER_UPDATE_SHOWBUYS’
    b. Datatype: Boolean
17. updateShowBuysReviews(showBuysReviews)
    a. Action type: ‘USER_UPDATE_SHOWBUYSREVIEWS’
    b. Datatype: Boolean
18. updateMailingAddress(mailAddress)
    a. Action type: ‘USER_UPDATE_MAILADDRESS’
    b. Datatype: String
19. updateZipCode(zipCode)

a. Action type: ‘USER_UPDATE_ZIPCODE’
b. Datatype: Integer

ii. INITIAL_BUYER_SETTINGS_STATE
firstName; datatype String
lastName; datatype String
birthday; datatype Date
email; datatype Integer
phone; datatype Character
userName; datatype String
password; datatype String
cardNumber; datatype Integer
cardExpiration; datatype Date
updateCVV; datatype Integer
updatePostalCode; datatype Integer
bioDescription: datatype String
rateStars: datatype Integer
review: datatype String
socialMedia; datatype String
showBuys: datatype Boolean
showBuysReviews: datatype Boolean
socialMedia: datatype String
showBuys; datatype Boolean
showBuysReviews; datatype Boolean
mailAddress: datatype String
zipCode; datatype Integer

11. Search data definitions
    a. Query; datatype URL object
    b. searchQuery/setSearchQuery(); datatype URL object/empty String
    c. filterProducts(products, query); datatype Array

2. Functional Requirements

Priority 1

Marketplace

1. Unregistered and registered users shall be able to query the database for products by interacting with a search bar
2. Unregistered and registered users shall be able to access ‘Contact’ and ‘About us’ pages of the website.
3. Unregistered and registered users shall be able to get access to technical support.
4. Sellers shall be able to put their products up for auction or list them in the marketplace
5. Buyers shall be able to propose buy it now or best offer options for a seller’s product
6. Buyers shall be able to filter the products they are searching for, based on minimum/maximum price, location filtering, or type of shipping (pickup/delivery)
7. Buyers shall be able to check their account settings to determine if a product they’ve purchased has been confirmed, processed, shipped, or returned
8. Buyers shall be given a tracking number to stay up to date with their product’s delivery status

Website Features

9. Sellers shall be provided with consignment operations to get their products delivered to buyers
10. Buyers shall be given a price matching tool, which compares products either already posted on our app or compares other products from other websites (amazon)

Administrators

11. Administrators shall be able to change buyer/seller usernames and passwords.
12. Administrators shall be able to modify account settings for buyer/seller.
13. Administrators shall be able to delete products from the admin page
14. Administrators shall be able to delete accounts.
15. Administrators shall be able to see all product discussions for a specific product
16. Administrators shall be able to delete product discussions for a specific product
17. Administrators shall be able to delete price matching products
18. Administrators shall be able to modify Auction settings.
19. Administrators shall be able to search for a specific user
20. Administrators shall be able to view all data related to a specific user
21. Administrators shall be able to see the shopping cart data of a specific user
22. Administrators shall be able to ban a user from Dropsell
23. Administrators shall be able to restrict certain features of Dropsell from a specific user
24. Administrators shall have a live chat feature they can join to communicate directly with users

Buyers

25. Buyers shall sign a purchase agreement; they must agree to conduct business according to Dropsell's rules prior to using the marketplace
26. Buyers shall be able to send inquiries of interest to a product's seller
27. Buyers shall save products they wish to purchase by adding them into a shopping cart
28. Buyers shall be able to remove an item from their shopping cart and the total list of products should be updated accordingly
29. Buyers shall be able to return to the main shopping menu to look for other products if they are not finished shopping
30. Buyers shall have the option of canceling or modifying their order
31. Buyers shall receive a detailed receipt of their purchase through email after checking out
32. Buyers shall be able to add their full name, email, phone number, and delivery address if they haven't inputted this data prior to checking out
33. Buyers shall have the option of choosing an existing payment option or adding a new one
34. Buyers shall be notified by final invoice of their purchase's pickup/delivery time, expected time of arrival, name of seller, location to meet them at, their contact info, and the grand total to pay

Sellers

35. Sellers shall sign a contract before being granted the privilege of posting products on Dropsell
36. Sellers shall agree to a small listing fee for each product they sell
37. Sellers shall be able to create new products with a title, description, price, category, and image
38. Sellers shall be able to edit the title, description, price, category, and image of their products
39. Sellers shall be able to adjust the quantity of a product they've listed on Dropsell
40. Sellers shall be able to delete products they've listed
41. Sellers shall be shown all their listed products under the seller settings section of their profile

Auction

42. Sellers shall have full control over their auction settings, from choosing the average starting bid, the time when the auction begins, to the total duration of the auction
43. Sellers shall be able to set the duration of their auction for as long as 30 days or as short as 1 hour
44. Buyers shall be displayed the current price of an auction, the amount of time remaining, and any bids from other buyers
45. Buyers shall be able to bid on products with one click.
46. Buyers and sellers shall see a live countdown of the auction's remaining time

Messaging

47. Buyers shall have the ability to contact sellers directly through the Dropsell website
48. Buyers shall have the ability to contact sellers via email or phone, if they choose to do so

Priority 2

Marketplace

49. Buyers shall be given an advanced search bar for detailed searches, including product sizes, price ranges, and colors
50. Buyers shall be refunded in full if they purchase multiple products which causes the product to be sold out
51. Products shall have their reviews displayed when listed in the marketplace
52. Products shall be displayed as out of stock if the supply has run out
53. Sellers shall be able to publish different styles of their products, such as different colors or fabrics
54. Unregistered and registered users shall be able to share products with friends through email, social media, or a shareable link
55. Registered users shall have the option of flagging products as inappropriate, a scam, or submit a ticket to the Dropsell team for further review
56. Buyers and sellers shall be able to schedule and edit meetup times
57. Users should be offered a base price for automatic reject or accept
58. Buyers shall be able to access the location information of sellers through an interactive map
59. Unregistered and registered users shall be shown ads of products similar to their search histories
60. Buyers shall be given a clear breakdown of shipping options to choose from
61. Buyers shall be contacted by email if they opt to be notified when a product comes back in stock
62. Unregistered and registered users shall be able to go back to products they have viewed previously on their user feed
63. The marketplace shall show the maximum time length for shipping a product
64. The marketplace shall show the minimum time length for shipping a product
65. The marketplace shall perform monitoring on unregistered and registered users
66. The marketplace shall keep track of the ID’s of posts unregistered and registered users click on which will be saved in the database under most-interacted products
67. The marketplace shall contain a currency converter for entering international markets

Website Features

68. Sellers shall be able to see statistics related to how many buyers have purchased their products
69. Sellers shall be provided with an algorithm which keeps track of how many interactions their products have
70. The checkout page shall calculate tax automatically based on the buyer’s location
71. Buyers shall be provided with daily deals on the home page
72. Sellers shall be provided with a profitability test for determining how much it costs to ship a product on Dropsell as opposed to other ecommerce services
73. Users shall be able to zoom in or out of product images with a magnifying glass feature
74. Sellers shall be provided with a daily product forecast

75. Products shall contain hashtags to help with categorization
76. Unregistered and registered users shall be able to search products using hashtags
77. Buyers shall be shown products which are similar in price or category to the product they are currently viewing
78. Buyers shall be shown comparisons between product specifications
79. Sellers shall be able to keep track of the median price ranges of products they are viewing

Buyers

80. Buyers shall receive shipping information through email if their purchase involves shipping products to their specified address
81. Buyers shall be able to rate and star products
82. Buyers shall be able to interact with all of their starred products in their buyer settings
83. Buyers shall be able to choose from various UPS delivery methods
84. Buyers shall be able to view all data related to a seller's review page
85. Buyers shall add auction products to their watch list to keep track of the auction's most recent updates
86. Top-rated products shall be displayed to buyers after they purchase a similar product
87. Buyers shall be permitted to submit reviews of products they purchase
88. Buyers shall be able to click on product listing images to zoom in and see more details
89. Buyers shall be provided with a list of their purchase histories
90. Buyers shall be able to save products they'd like to buy in the future to a wishlist
91. Buyers shall be rewarded with discount codes if they are frequent buyers
92. Buyers shall be provided with a promotion code box for entering their discount code when checking out
93. Buyers shall be able to generate a referral URL which they can send to friends
94. Buyers shall be able to subscribe to a specific seller so they can be notified when a product's price is updated, the product has been restocked, or if the product has been removed
95. Buyers shall be notified about products they have viewed previously
96. Buyers shall be asked if they are still interested in the products they've previously viewed

Sellers

97. Sellers shall get an email confirmation after one of their products has sold y
98. Sellers shall allow their data such as user details, products, profile picture, to be publicly displayed to buyers
99. Sellers shall have the option of selecting multiple products to delete
100. Sellers shall have the option of relisting their products for sale
101. Sellers shall be able to advertise their products to specific buyers with a send offer feature
102. Seller products shall have a review page where buyers send feedback about its quality
103. Sellers shall have a review page where buyers can provide structured feedback
104. Sellers shall have a rating system based on a five-star system
105. Sellers shall be required to display "illegal item" on illegal products

106. Sellers shall be required to explain their illegal item and their reasons for listing it
107. Seller products shall be marked as out of stock if its supply runs out
108. Seller products shall be marked as “Last 1 Available” so buyers know that it is the last product remaining
109. Sellers shall participate in a striking system if they choose to perform misconduct or violate the terms of service.
110. Sellers shall have 3 strikes before their account faces a possible suspension or blacklist

Auction

111. Buyers shall not be able to retract a bid they place on a product
112. Buyers shall be notified immediately by email that they’ve won an auction
113. Buyers shall bid on products as many times as possible before the remaining time runs out
114. Buyers shall be able to keep track of auction statistics
115. Buyers shall be able to see how many other buyers are bidding on the same product
116. Buyers shall be notified with a 5 minute warning before an auction finishes
117. Buyers shall be notified when the auction finishes
118. Buyers shall be notified after they win an auction
119. Buyers shall have the option of choosing from multiple payment options, such as Paypal, ApplePay, or GooglePlay

Priority 3

Marketplace

120. Sellers shall be able to create a wedding registry with a product wishlist for their special day
121. Buyers shall be shown other buyers who have made the same purchases
122. Buyers and sellers shall have the time length of shipping a product hidden if their location is closer than 1 or 2 miles

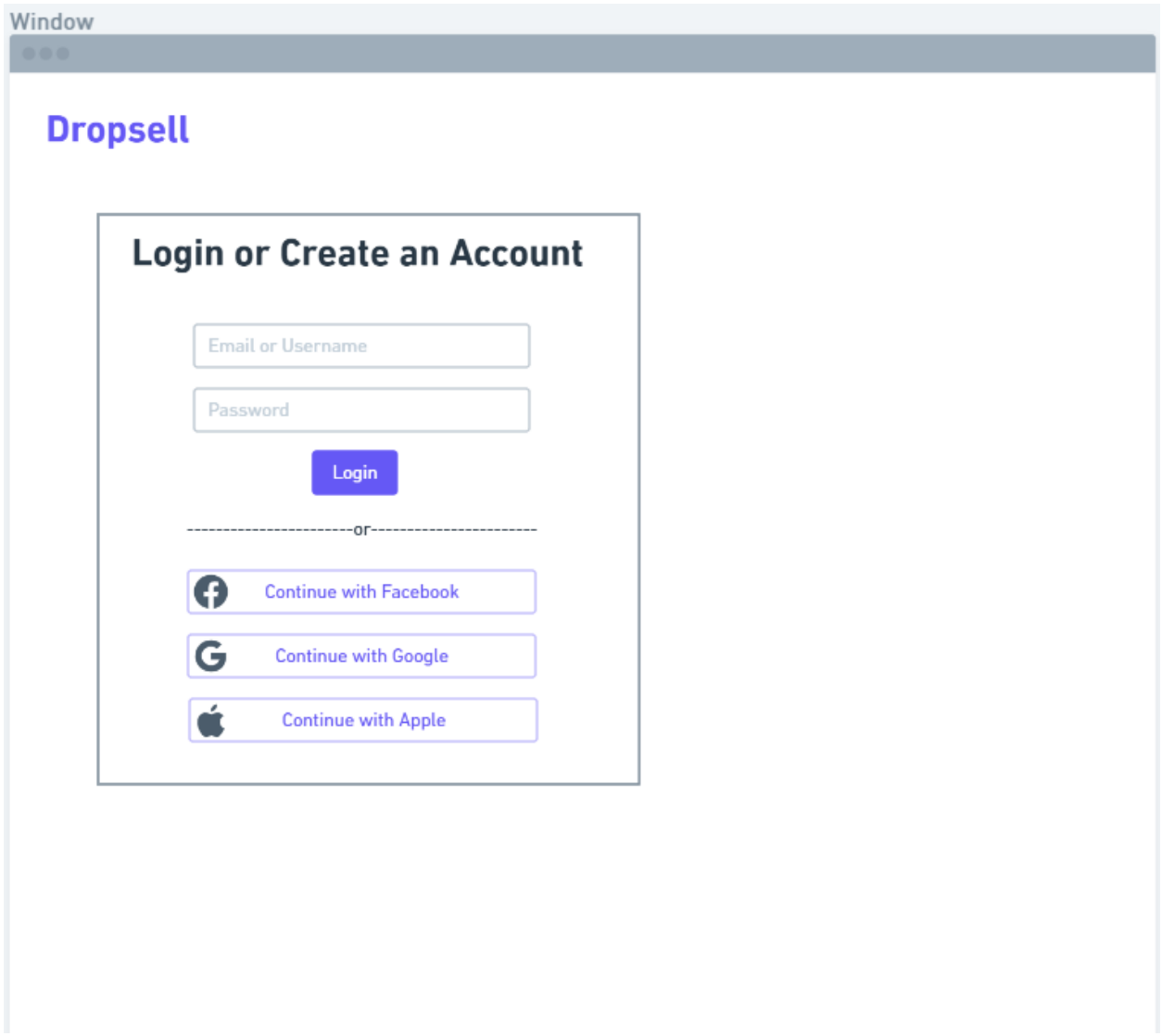
Buyers

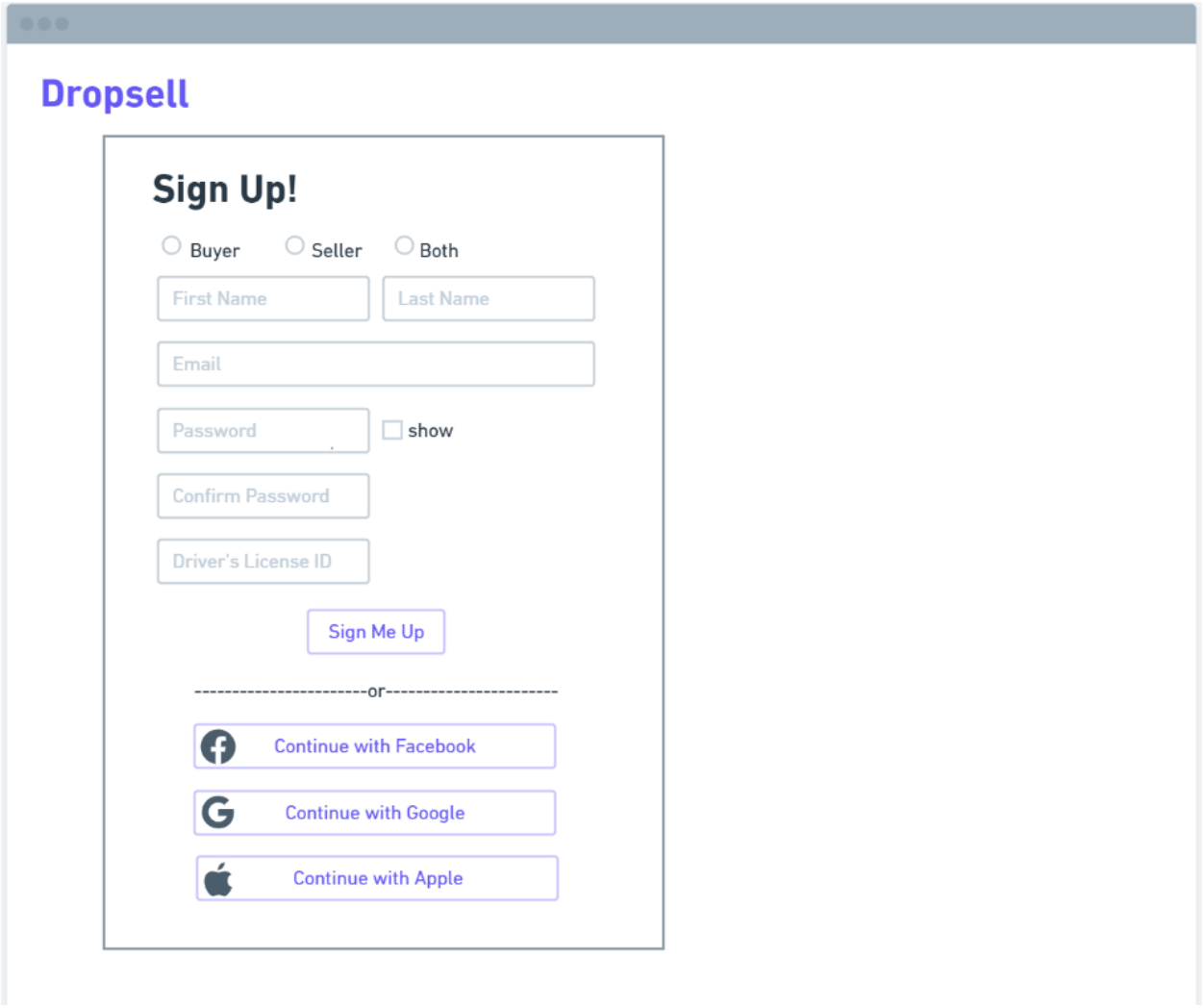
123. Buyers shall be awarded with a random product if they purchase products deemed as lucky
124. Buyers shall be rewarded with a promotion code or a random product for their birthday

Sellers

125. Sellers shall have background checks if they intend to sell on Dropsell
126. Sellers shall have their products hide ratings if they have 5 stars

3. Wireframes Based on Mockups/Storyboards




Dropsell
Sign Up!
Buyer
Seller
Both
First Name
Last Name
Email
Password
show
Confirm Password
Driver's License ID
Sign Me Up
----------or----------
Continue with Facebook
Continue with Google
Continue with Apple

Dropsell

Starred/Watching

BNIB Wireless Dualshock PS4 Controller

Condition: New
Quantity: 3
Price: $40.00

BUY IT NOW

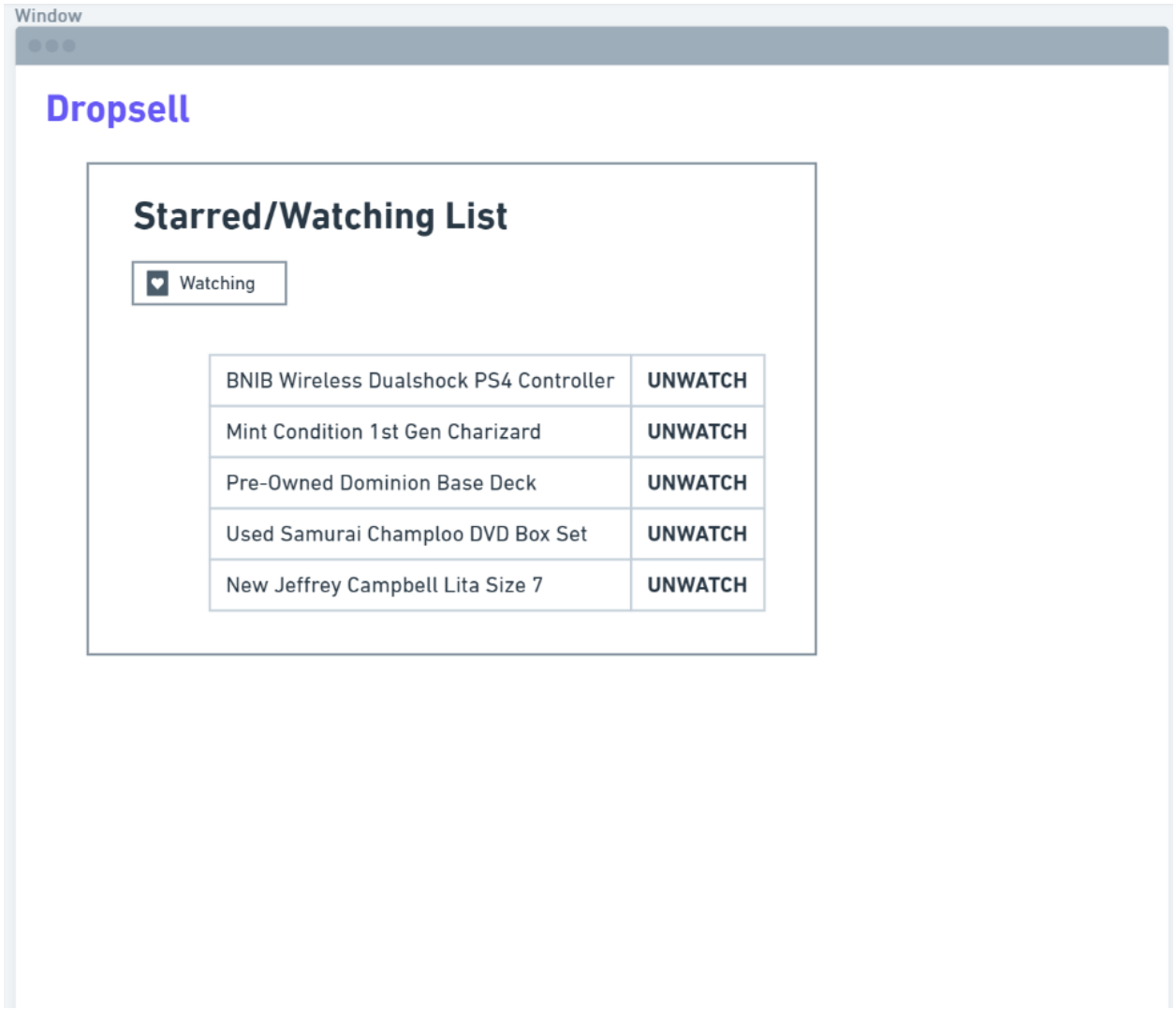
ADD TO CART

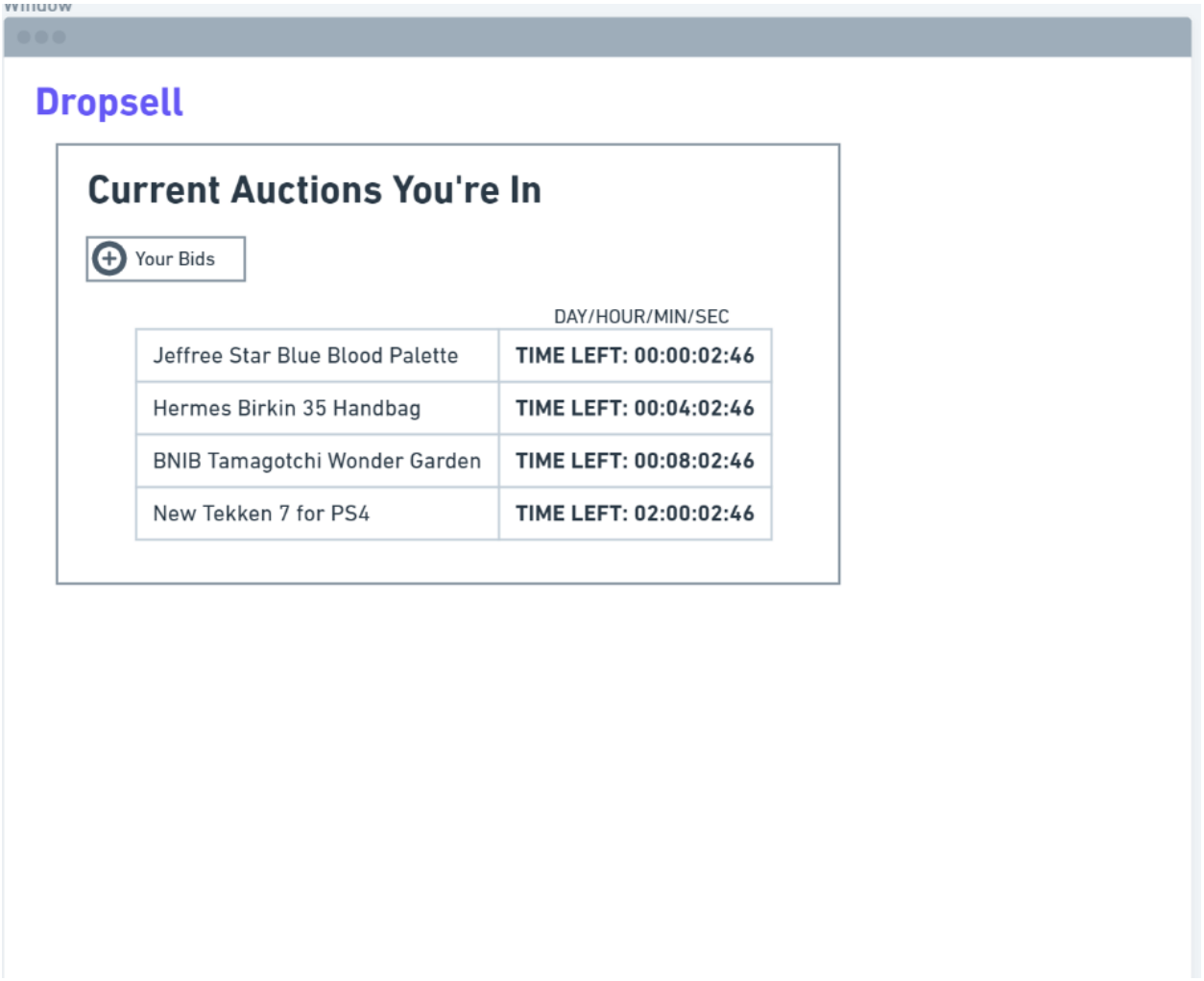
WATCHING

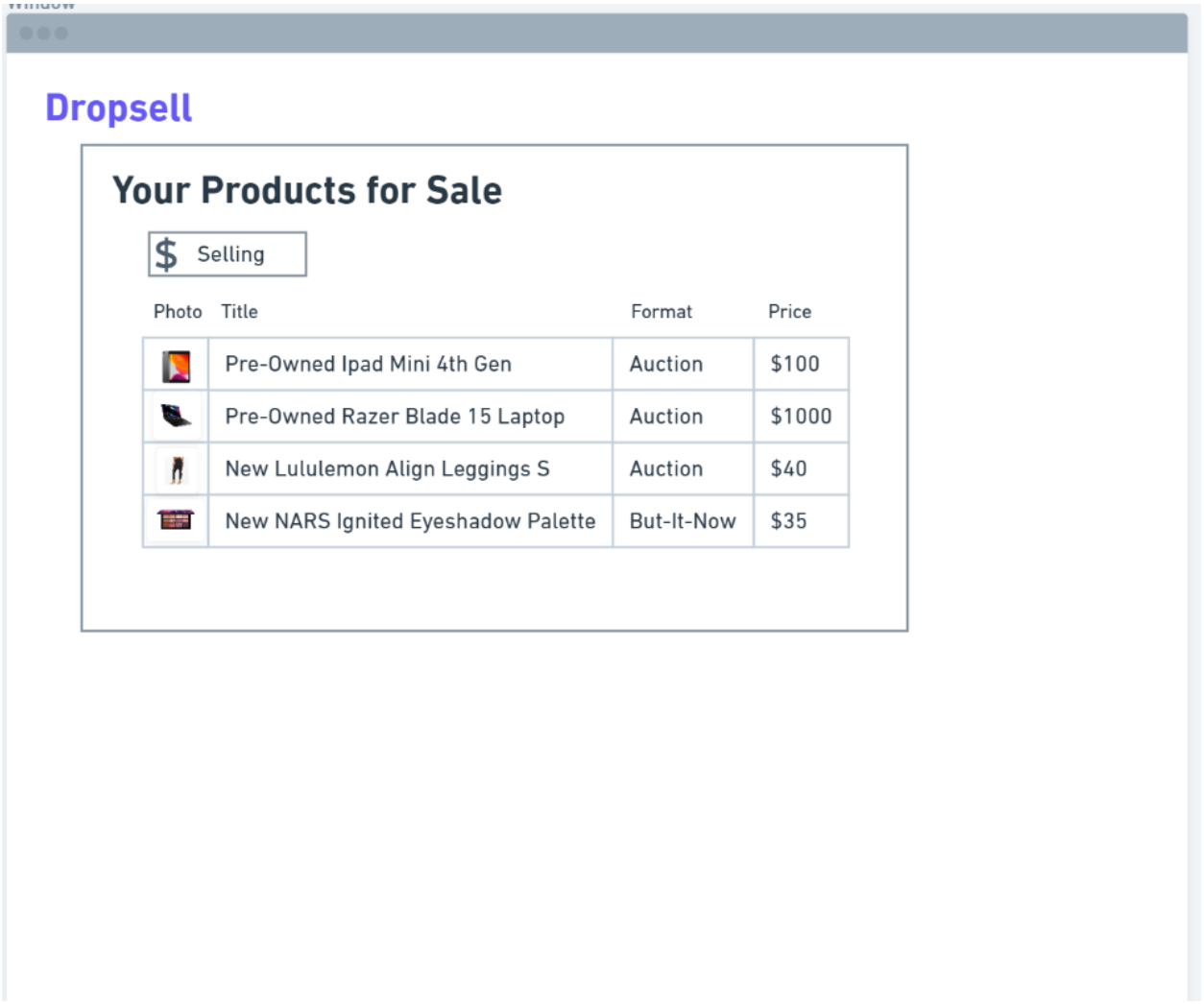
Shipping: FREE
Delivery: Jul 12- Jul 20
Payments: Paypal

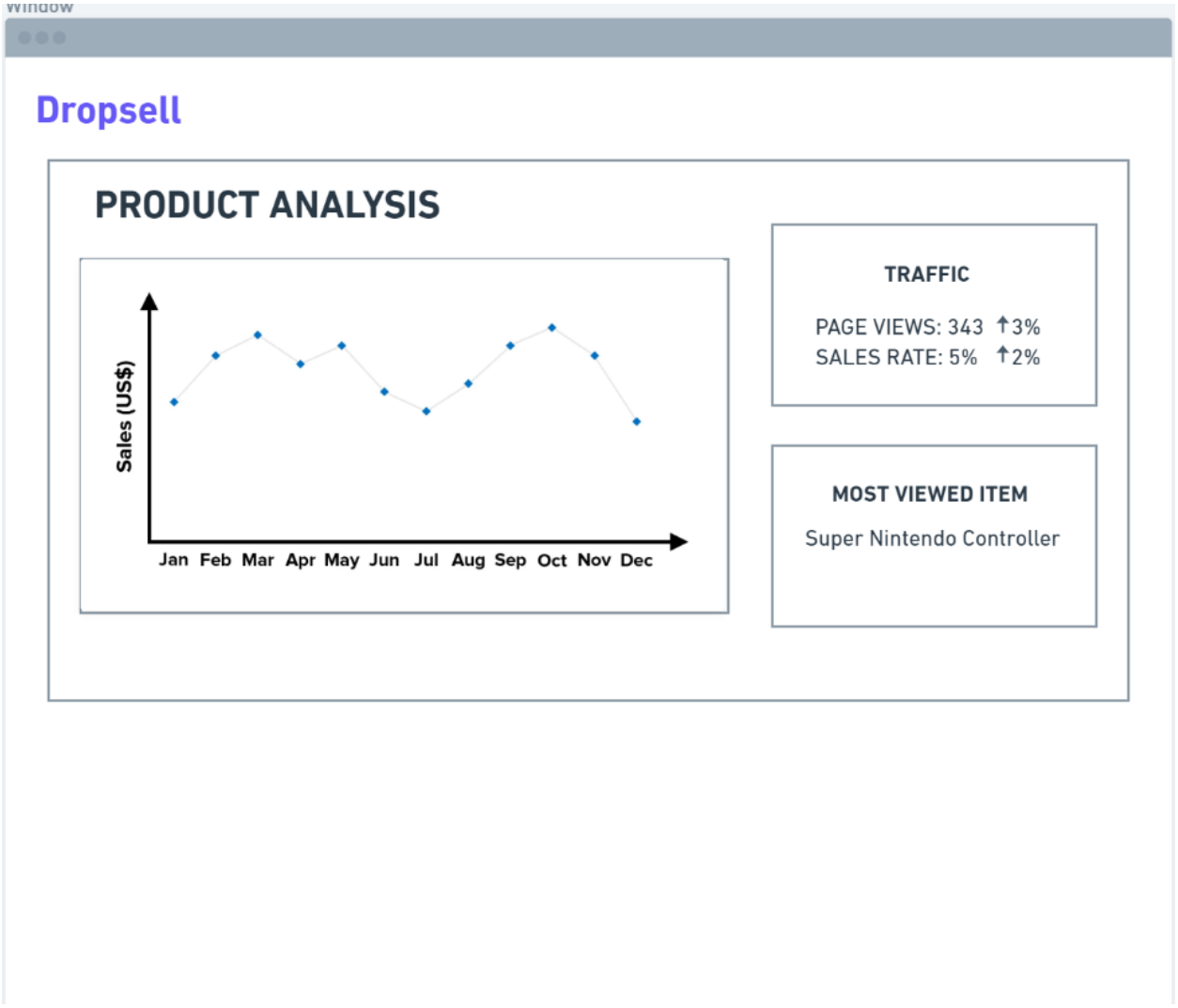
Seller: Jane Doe

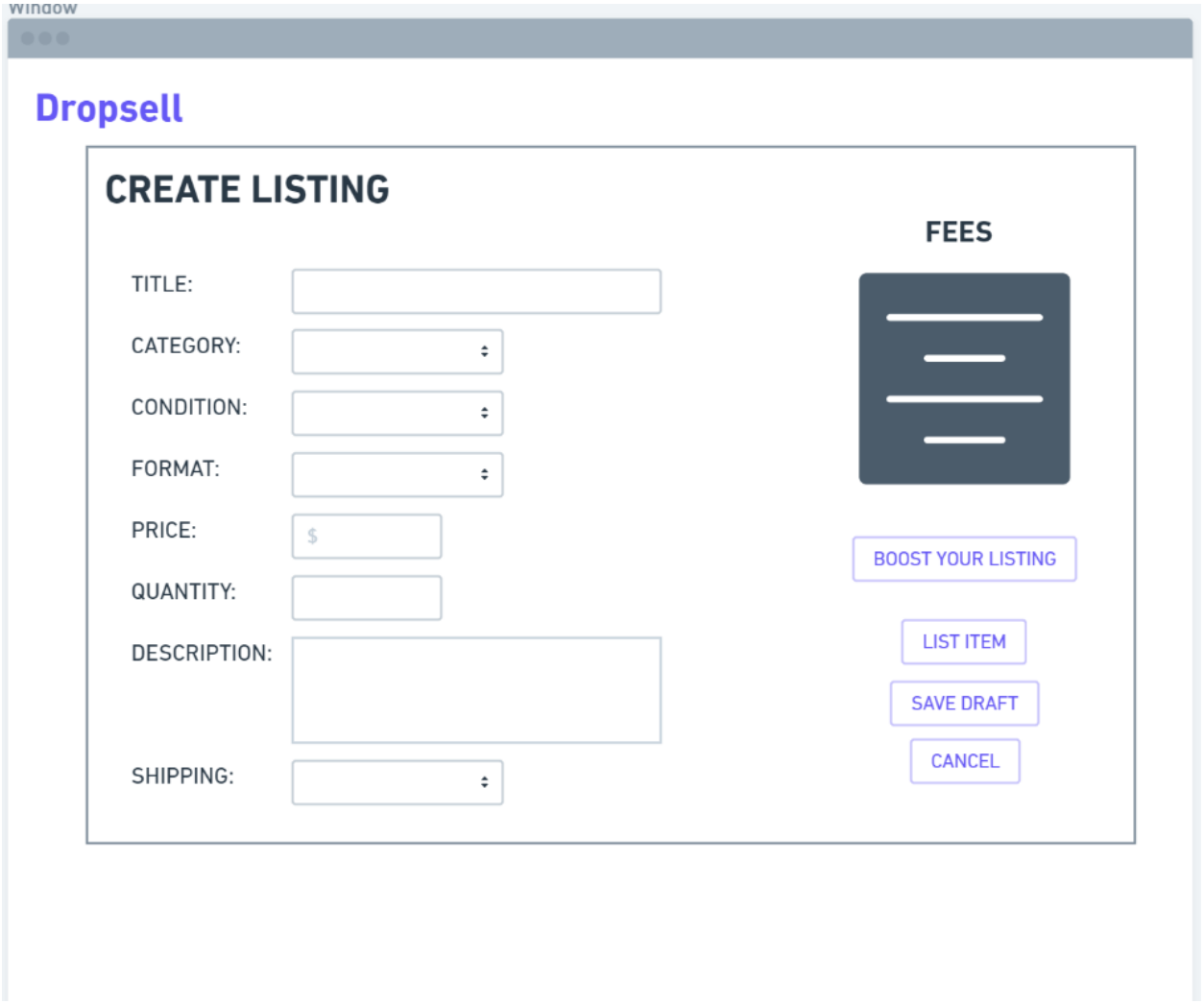
CONTACT SELLER

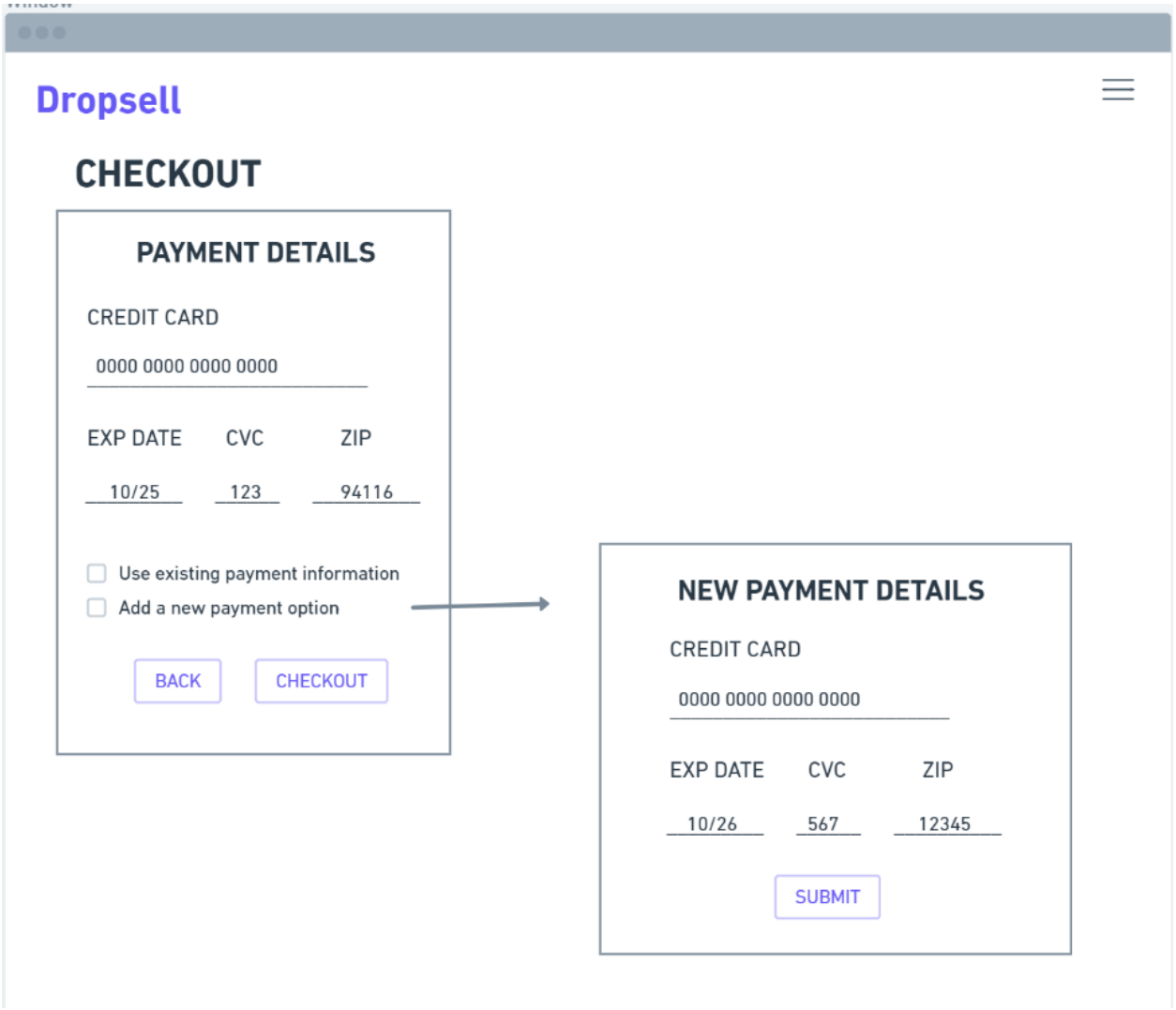

Window
Dropsell
Starred/Watching List
Watching
BNIB Wireless Dualshock PS4 Controller
UNWATCH
Mint Condition 1st Gen Charizard
UNWATCH
Pre-Owned Dominion Base Deck
UNWATCH
Used Samurai Champloo DVD Box Set
UNWATCH
New Jeffrey Campbell Lita Size 7
UNWATCH

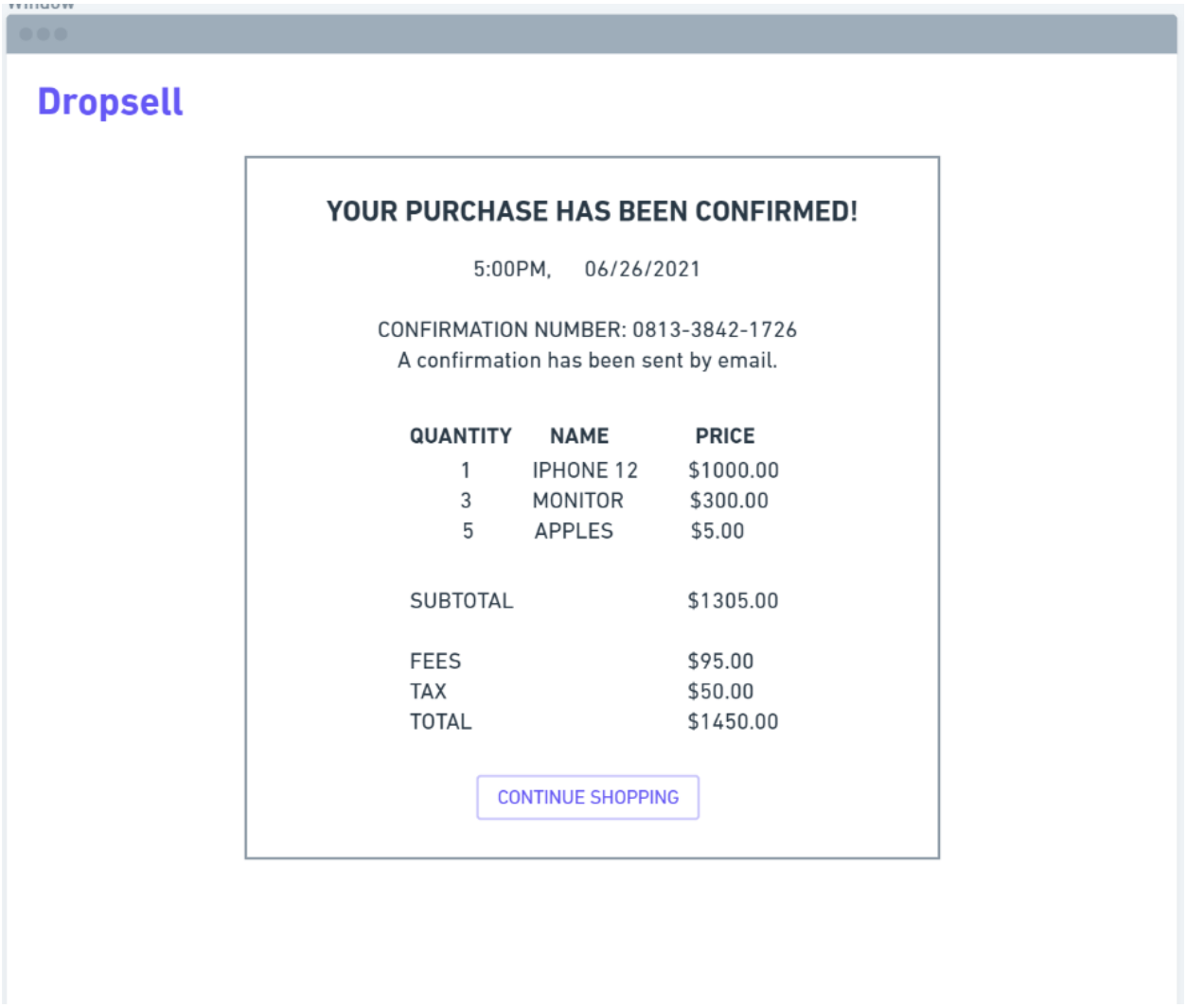

Window
Dropsell
Current Auctions You're In
Your Bids
DAY/HOUR/MIN/SEC
Jeffree Star Blue Blood Palette
TIME LEFT: 00:00:02:46
Hermes Birkin 35 Handbag
TIME LEFT: 00:04:02:46
BNIB Tamagotchi Wonder Garden
TIME LEFT: 00:08:02:46
New Tekken 7 for PS4
TIME LEFT: 02:00:02:46

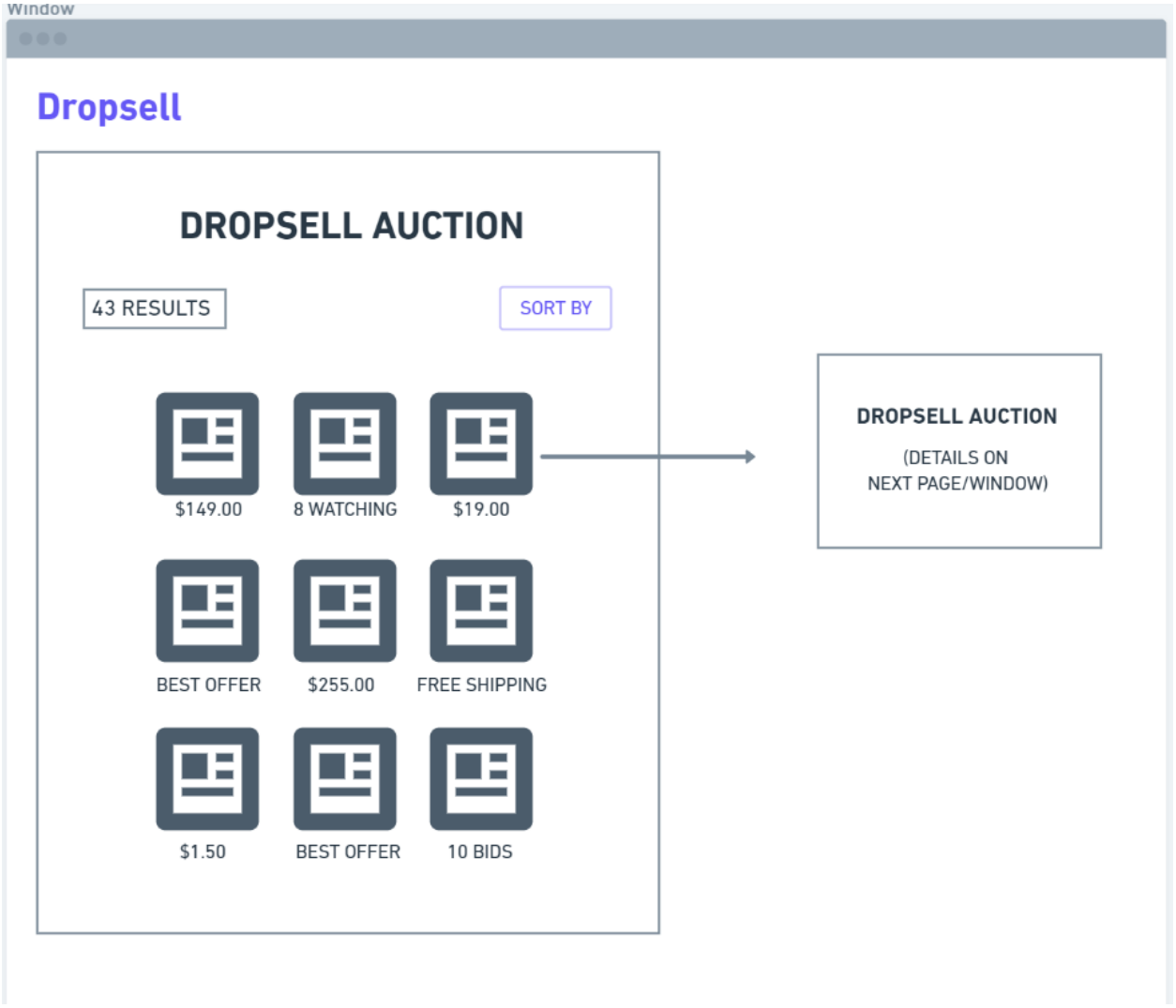

Dropsell
Your Products for Sale
Selling
Photo
Title
Format
Price
Pre-Owned Ipad Mini 4th Gen
Auction
$100
Pre-Owned Razer Blade 15 Laptop
Auction
$1000
New Lululemon Align Leggings S
Auction
$40
New NARS Ignited Eyeshadow Palette
But-It-Now
$35

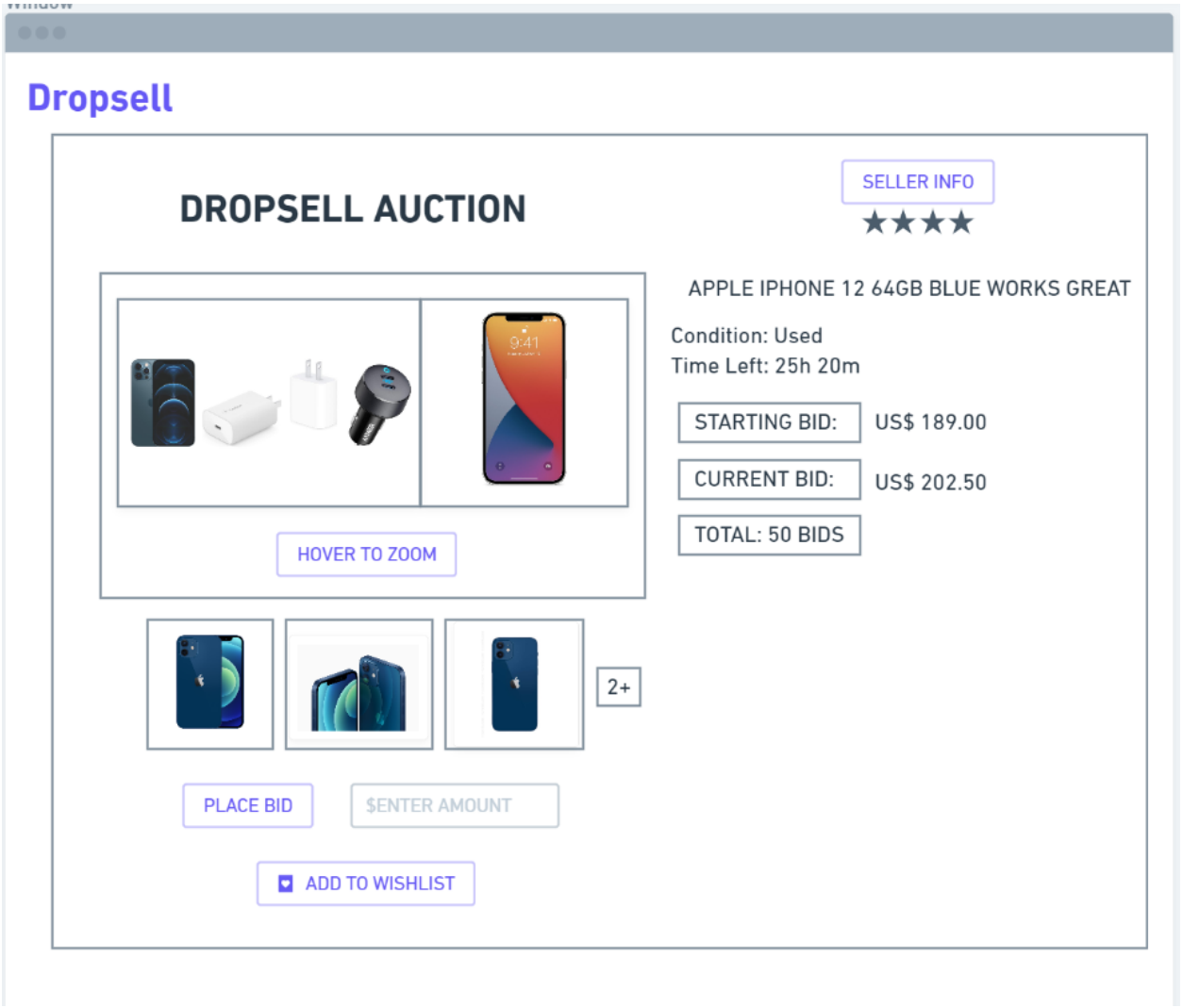

Window
Dropsell
PRODUCT ANALYSIS
Sales (US$)
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
TRAFFIC
PAGE VIEWS: 343 ↑3%
SALES RATE: 5% ↑2%
MOST VIEWED ITEM
Super Nintendo Controller


Window
Dropsell
CREATE LISTING
TITLE:
CATEGORY:
CONDITION:
FORMAT:
PRICE:
$
QUANTITY:
DESCRIPTION:
SHIPPING:
FEES
BOOST YOUR LISTING
LIST ITEM
SAVE DRAFT
CANCEL

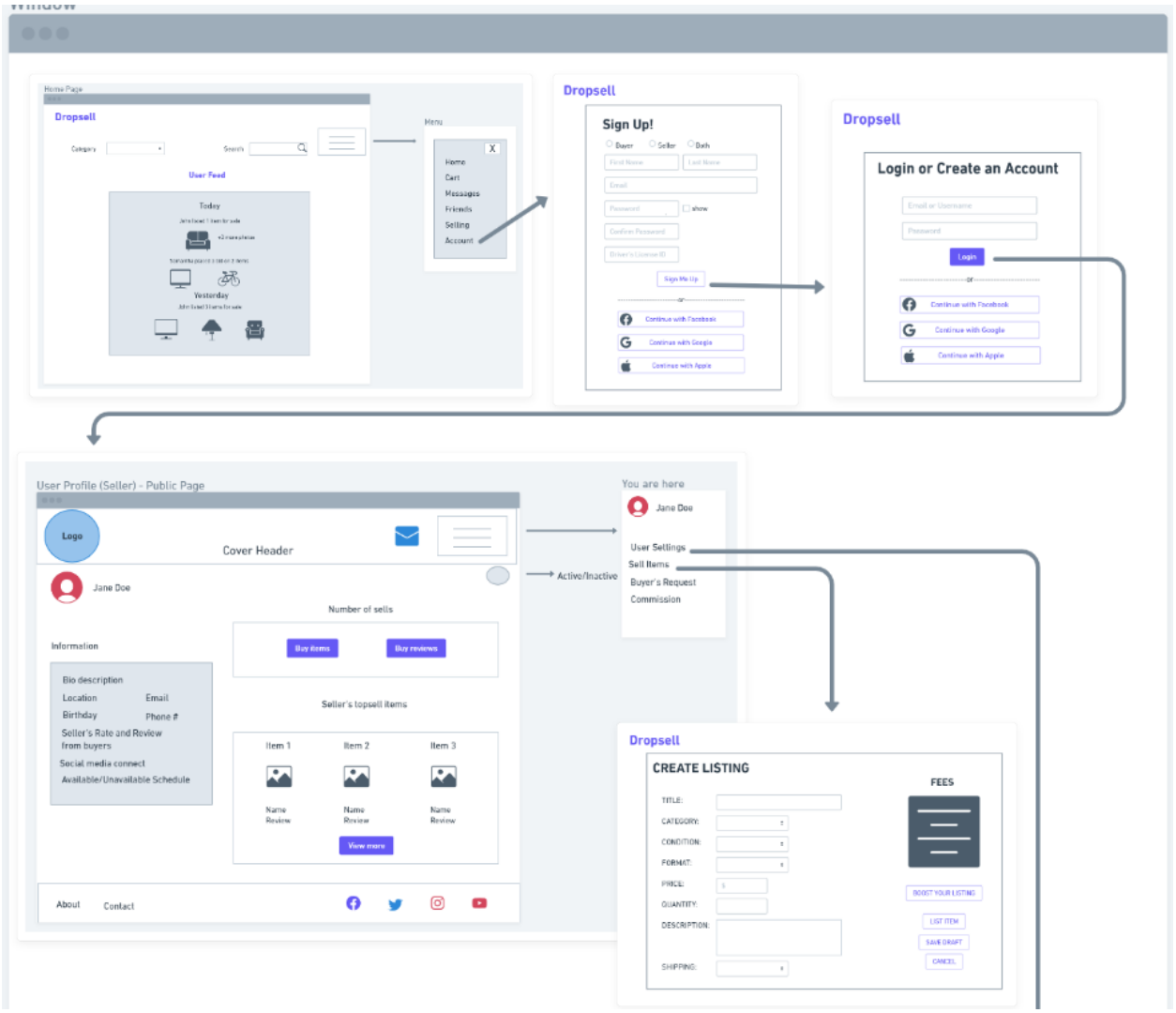

Dropsell
CHECKOUT
PAYMENT DETAILS
CREDIT CARD
0000 0000 0000 0000
EXP DATE
CVC
ZIP
10/25
123
94116
Use existing payment information
Add a new payment option
BACK
CHECKOUT
NEW PAYMENT DETAILS
CREDIT CARD
0000 0000 0000 0000
EXP DATE
CVC
ZIP
10/26
567
12345
SUBMIT

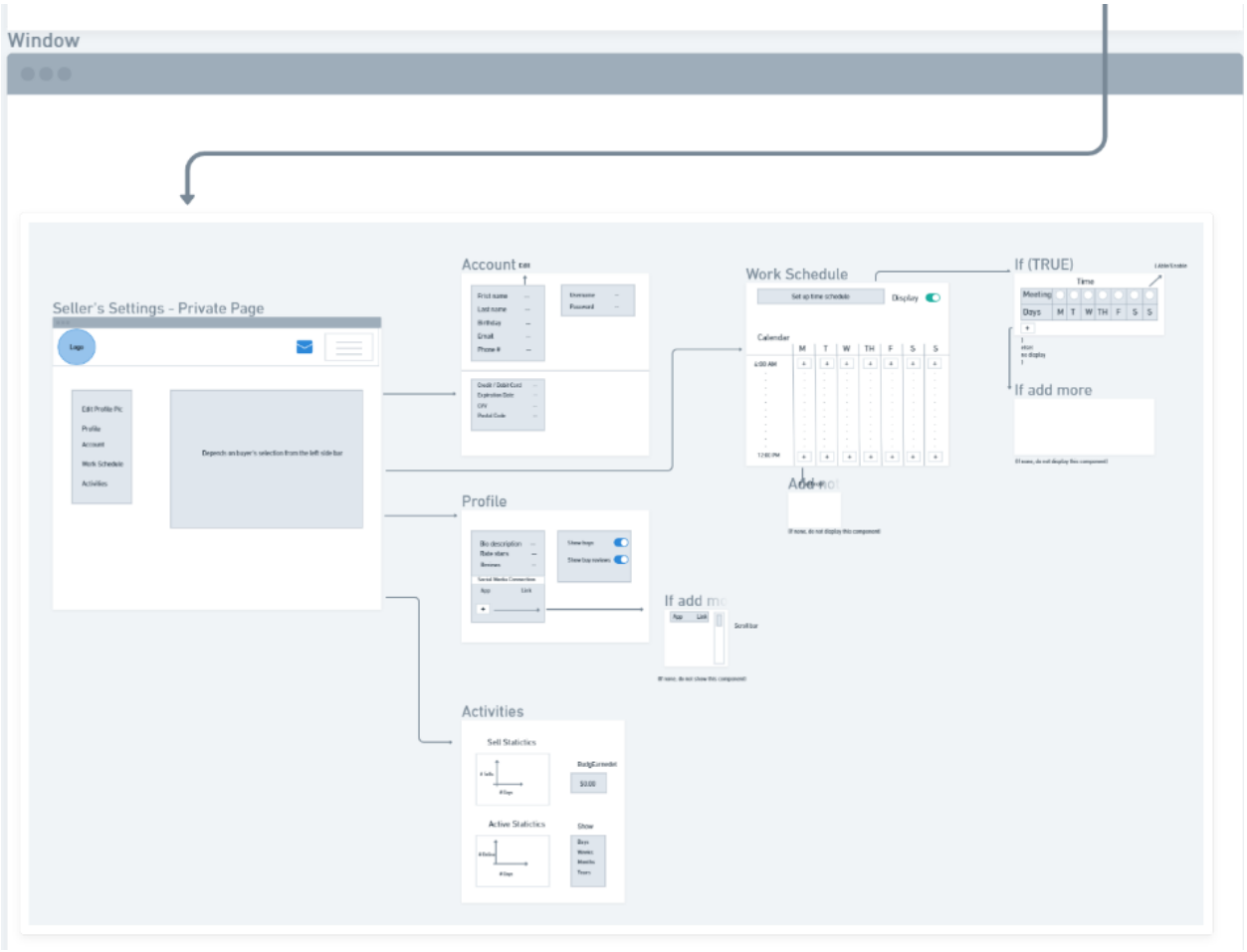

Dropsell
YOUR PURCHASE HAS BEEN CONFIRMED!
5:00PM, 06/26/2021
CONFIRMATION NUMBER: 0813-3842-1726
A confirmation has been sent by email.
QUANTITY NAME PRICE
1 IPHONE 12 $1000.00
3 MONITOR $300.00
5 APPLES $5.00
SUBTOTAL $1305.00
FEES $95.00
TAX $50.00
TOTAL $1450.00
CONTINUE SHOPPING


Window
Dropsell
DROPSELL AUCTION
43 RESULTS
SORT BY
$149.00
8 WATCHING
$19.00
BEST OFFER
$255.00
FREE SHIPPING
$1.50
BEST OFFER
10 BIDS
DROPSELL AUCTION
(DETAILS ON
NEXT PAGE/WINDOW)

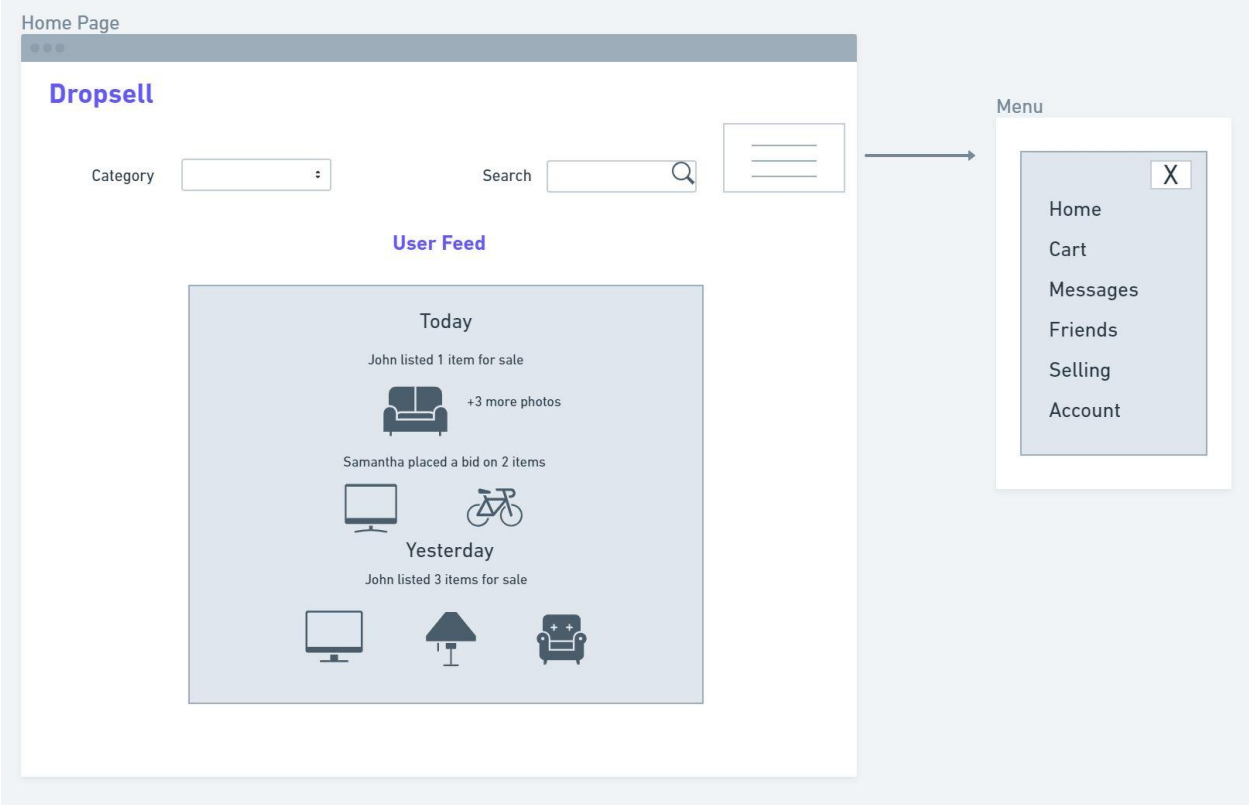

Dropsell
DROPSELL AUCTION
SELLER INFO
APPLE IPHONE 12 64GB BLUE WORKS GREAT
Condition: Used
Time Left: 25h 20m
STARTING BID:
US$ 189.00
CURRENT BID:
US$ 202.50
TOTAL: 50 BIDS
HOVER TO ZOOM
2+
PLACE BID
$ENTER AMOUNT
ADD TO WISHLIST

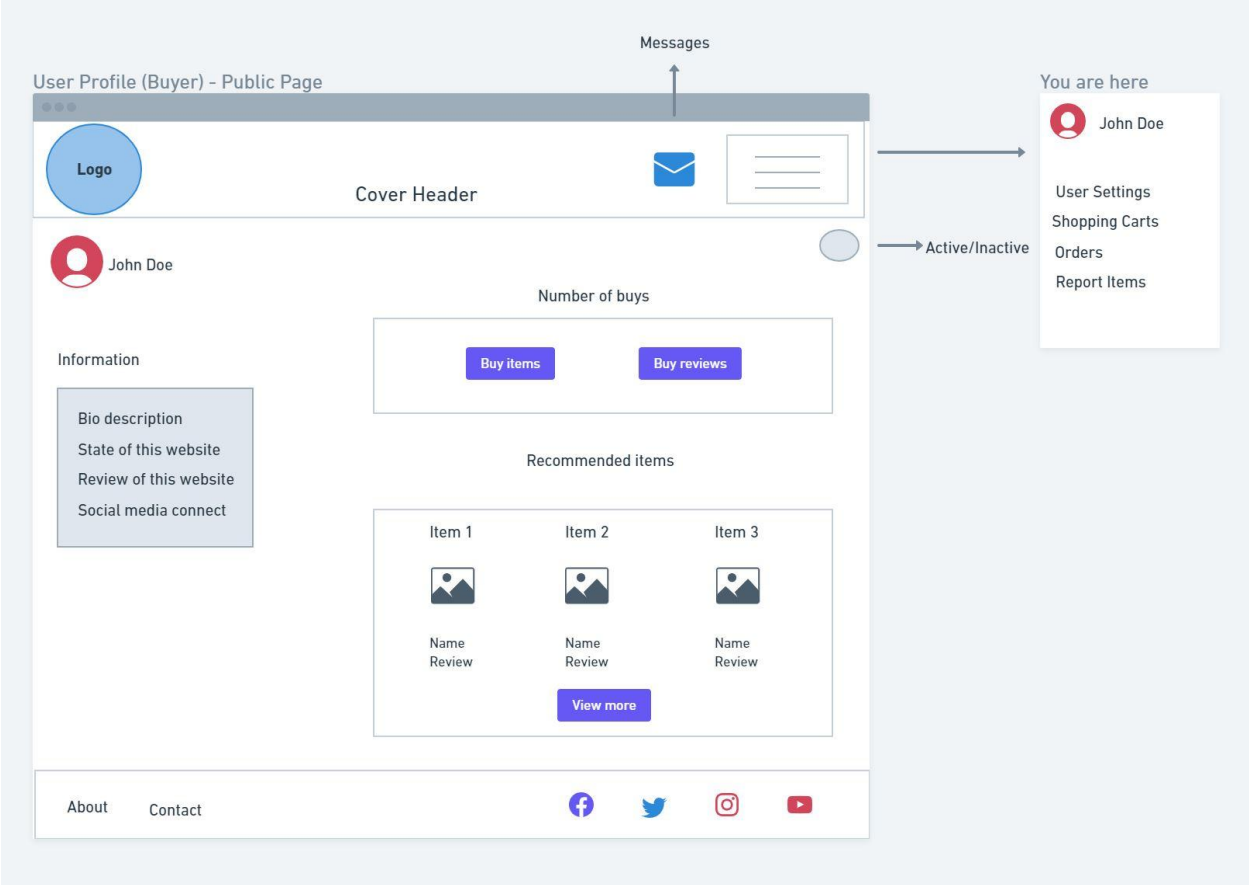
login -> register -> profile -> seller settings -> create product

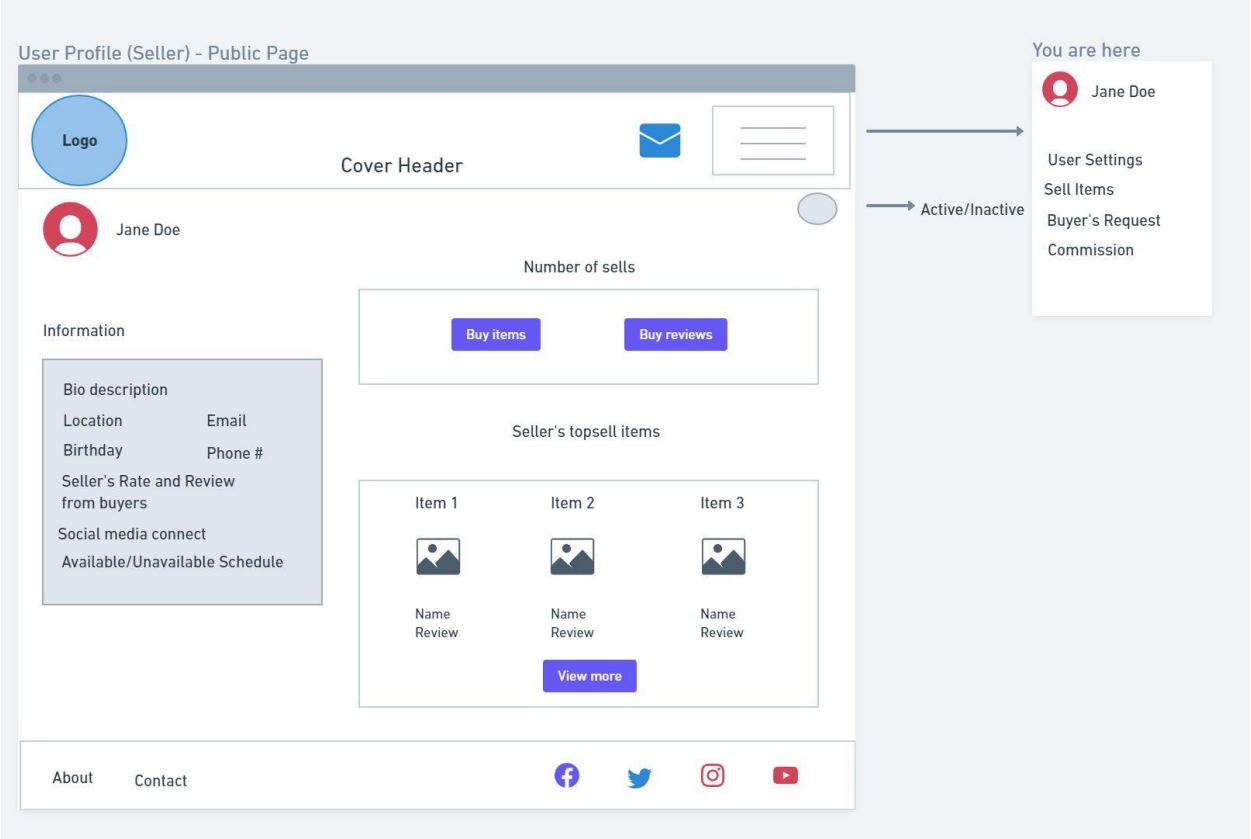


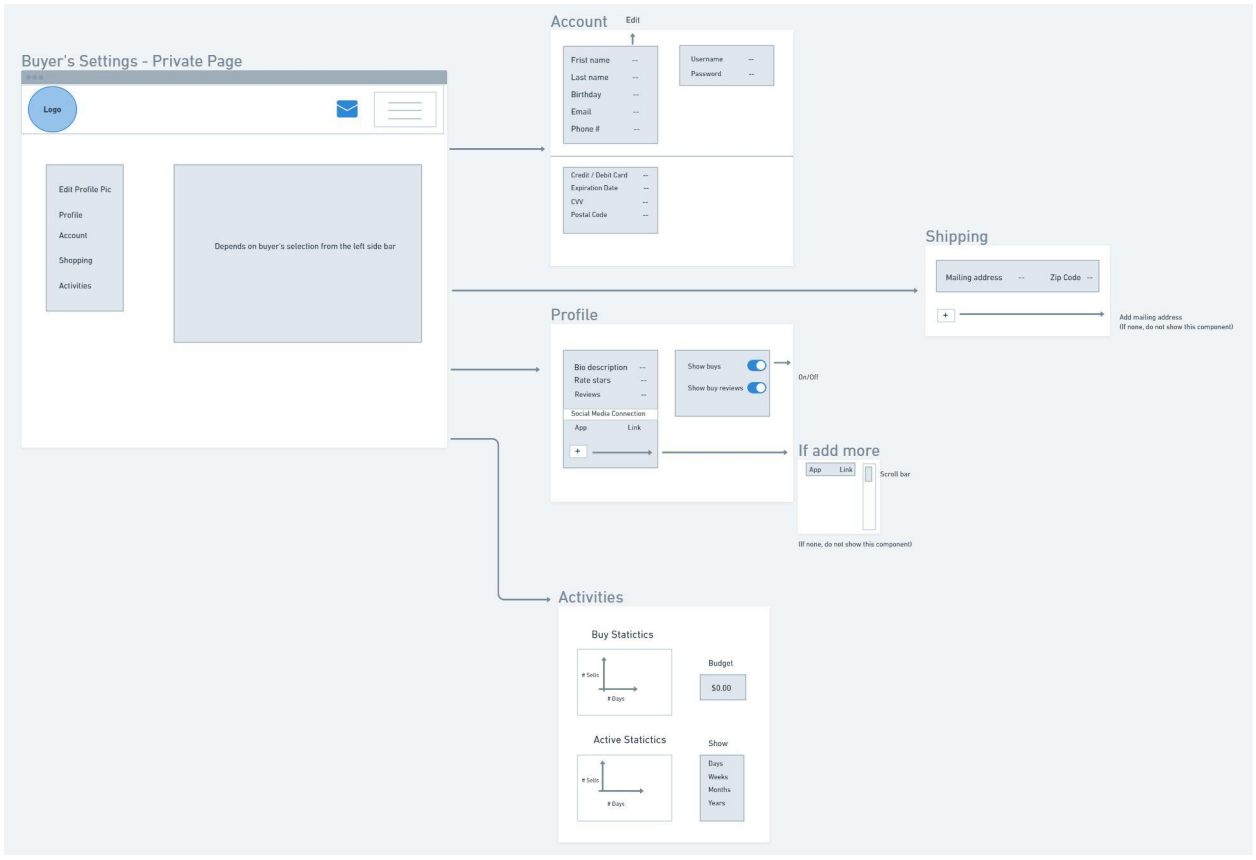

Window
Seller's Settings - Private Page
Account
Work Schedule
If (TRUE)
If add more
Profile
Activities
Sell Statictics
Active Statictics

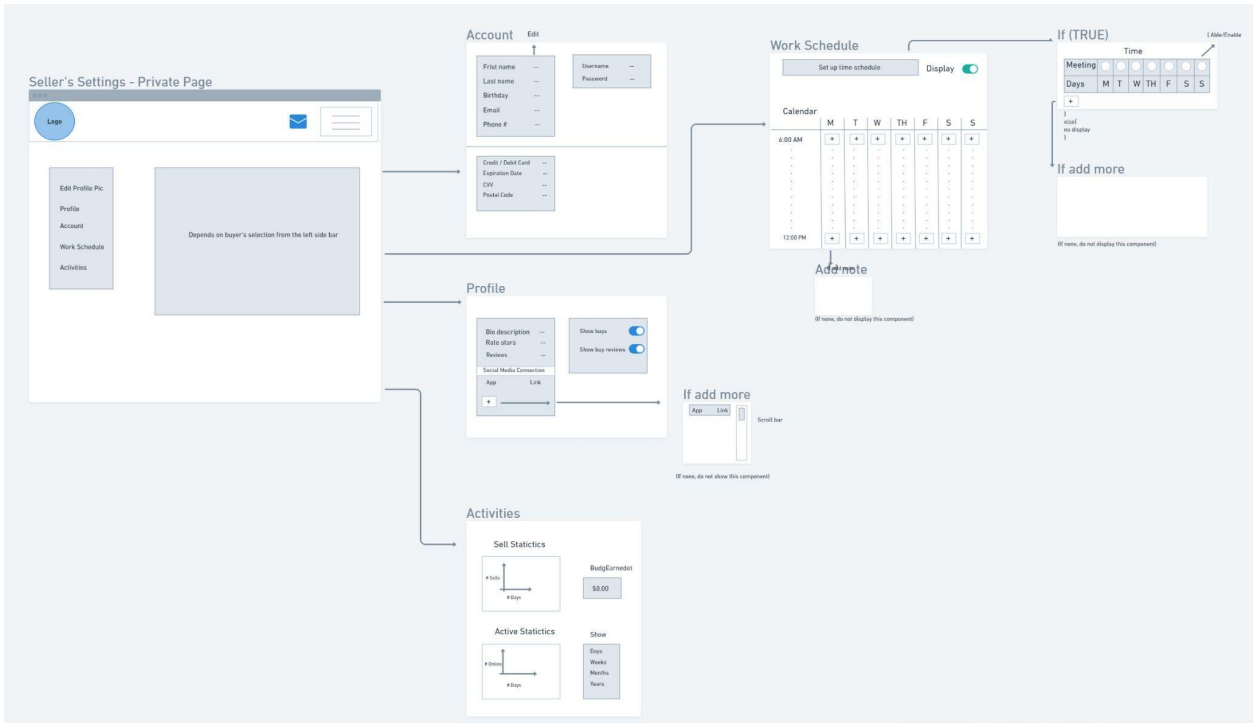

About
Logo
About Dropsell
Dropsell is a new digital marketplace created by seven seior students of San Francisco State University. We focus on our customers' safe and secure selling and buying experience.
Dropsell provides all sellers and buyers a platform to make transactions locally and globally, with little to no fees.

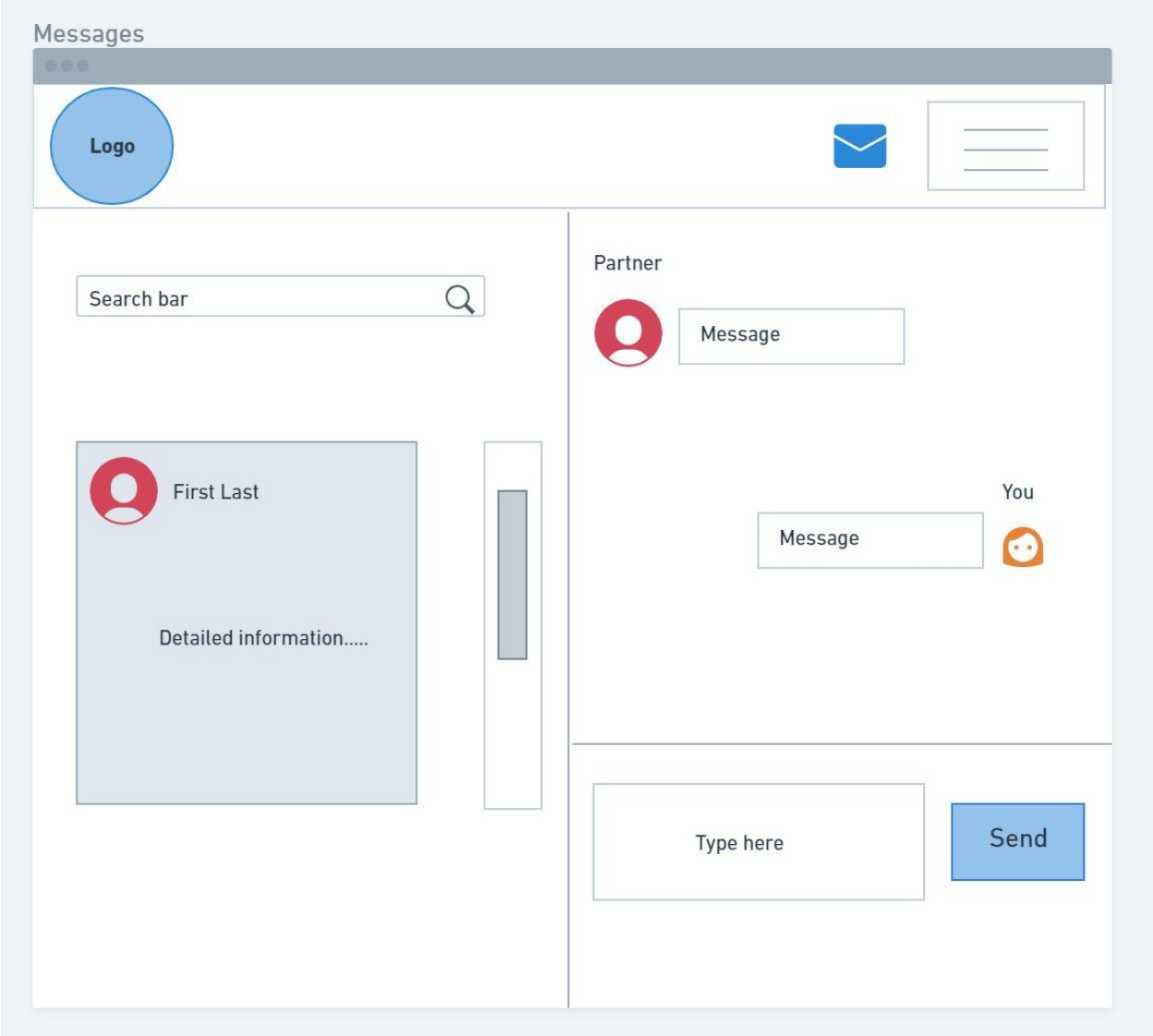

Home Page
Dropsell
Category
Search
User Feed
Today
John listed 1 item for sale
+3 more photos
Samantha placed a bid on 2 items
Yesterday
John listed 3 items for sale
Menu
X
Home
Cart
Messages
Friends
Selling
Account


Messages
User Profile (Buyer) - Public Page
You are here
Logo
Cover Header
John Doe
User Settings
Shopping Carts
Orders
Report Items
Active/Inactive
John Doe
Number of buys
Information
Buy items
Buy reviews
Bio description
State of this website
Review of this website
Social media connect
Recommended items
Item 1
Item 2
Item 3
Name
Review
Name
Review
Name
Review
View more
About
Contact


User Profile (Seller) - Public Page
Logo
Cover Header
Jane Doe
Number of sells
Buy items
Buy reviews
Information
Bio description
Location
Email
Birthday
Phone #
Seller's Rate and Review from buyers
Social media connect
Available/Unavailable Schedule
Seller's topsell items
Item 1
Item 2
Item 3
Name
Review
Name
Review
Name
Review
View more
About
Contact
You are here
Jane Doe
User Settings
Sell Items
Buyer's Request
Commission
Active/Inactive


Buyer's Settings - Private Page
Logo
Edit Profile Pic
Profile
Account
Shopping
Activities
Depends on buyer's selection from the left side bar
Account
Edit
Frist name --
Last name --
Birthday --
Email --
Phone # --
Username --
Password --
Credit / Debit Card --
Expiration Date --
CVV --
Postal Code --
Shipping
Mailing address -- Zip Code --
+
Add mailing address
(If none, do not show this component)
Profile
Bio description --
Rate stars --
Reviews --
Social Media Connection
App Link
+
Show buys
Show buy reviews
On/Off
If add more
App Link
Scroll bar
(If none, do not show this component)
Activities
Buy Statictics
Sells
Days
Budget
$0.00
Active Statictics
Show
Days
Weeks
Months
Years


Seller's Settings - Private Page
Logo
Edit Profile Pic
Profile
Account
Work Schedule
Activities
Depends on buyer's selection from the left side bar
Account
Edit
Frist name --
Last name --
Birthday --
Email --
Phone # --
Username --
Password --
Credit / Debit Card --
Expiration Date --
CVV --
Postal Code --
Work Schedule
Set up time schedule
Display
Calendar
M T W TH F S S
6:00 AM
12:00 PM
Add note
(If none, do not display this component)
If (TRUE)
Time
Meeting
Days M T W TH F S S
If add more
(If none, do not display this component)
Profile
Bio description --
Rate stars --
Reviews --
Social Media Connection
App Link
Show buys
Show buy reviews
If add more
App Link
Scroll bar
(If none, do not show this component)
Activities
Sell Statictics
Sells
Days
BudgEarnedet
$0.00
Active Statictics
Show
Days
Weeks
Months
Years


Messages
Logo
Search bar
First Last
Detailed information.....
Partner
Message
You
Message
Type here
Send

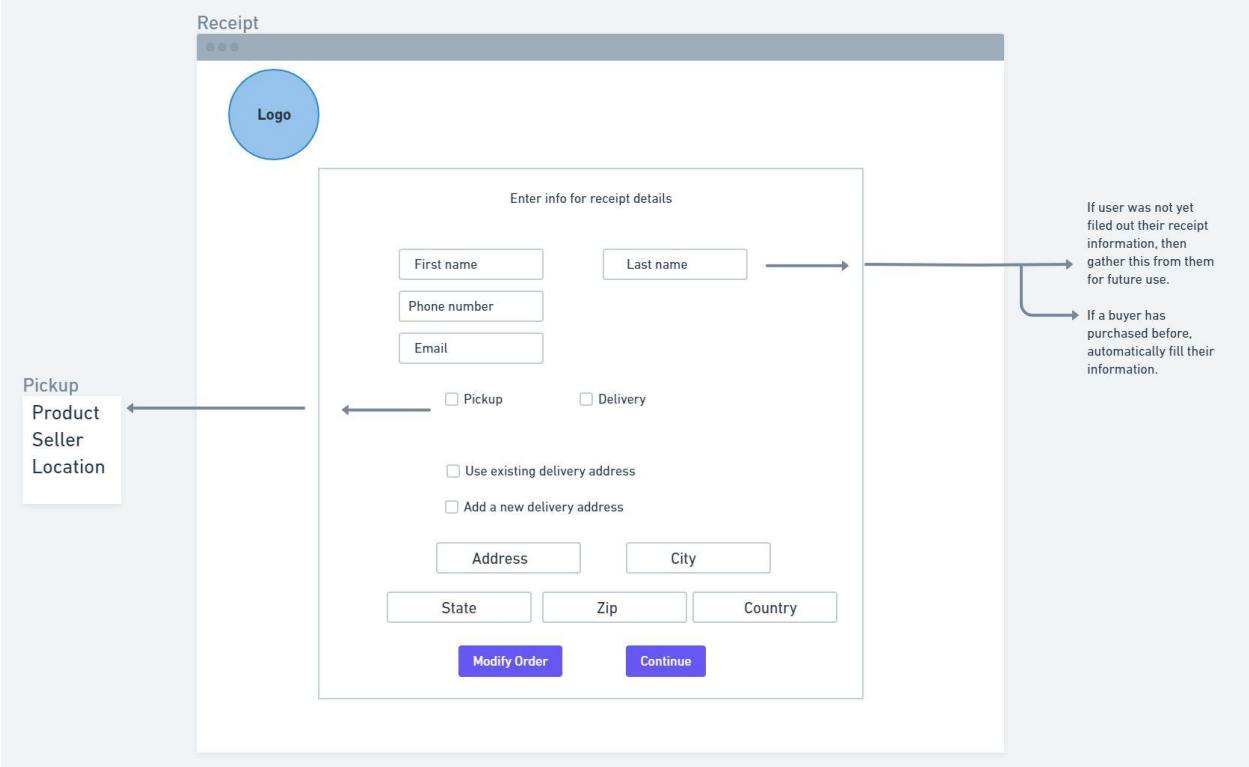
Summary

Logo

QTY	PRODUCT	NAME	PRICE
1		iPhone 12	$1000.00
3		Monitor	$300.00
4		Apples	$5.00

Subtotal	$1305.00
Fees	$95.00
Tax	$50.00
Total	$1450.00

Back Continue


Receipt
Logo
Enter info for receipt details
First name
Last name
Phone number
Email
Pickup
Delivery
Use existing delivery address
Add a new delivery address
Address
City
State
Zip
Country
Modify Order
Continue
If user was not yet filed out their receipt information, then gather this from them for future use.
If a buyer has purchased before, automatically fill their information.
Pickup
Product
Seller
Location

4. High Level Database Architecture and Organization

Business Rules

1. One registered user can create zero, one, or many products. A registered user shall upload at least one image, many images are optional.
2. One registered user can have zero, one, or many seller ratings. Registered users can optionally keep track of the rating data to display in seller analytics.
3. One registered user can have one session assigned to them. User sessions can optionally store session data related to the user, in addition to mandatory session id and expiration values.
4. One registered user can have one shopping cart. It is optional for registered users to add products into their shopping cart, it is not necessary for records to exist in a user's shopping cart before a user is registered.
5. Many registered users can create zero, one, or many product comments. Registered users can optionally edit or delete their comment data.
6. Many registered users can create zero, one, or many conversations. It is optional for registered users to have conversations with other registered users, they should still have full functionality of the website.

Entities, attributes, relationships, domains

1. Registered user
    a. Attributes
        i. user_id
        ii. username
        iii. email
        iv. password
        v. first_name
        vi. last_name
        vii. phone
        viii. street
        ix. city
        x. state
        xi. zip
        xii. is_active
        xiii. created_at
        xiv. is_buyer
        xv. is_seller
    b. Relationships (Other tables)
        i. sessions
        ii. shopping_cart
        iii. conversations, messages
        iv. meetups
        v. seller_ratings
        vi. products
2. Sessions
    a. Attributes
        i. session_id
        ii. session_expires
        iii. session_data
    b. Relationships

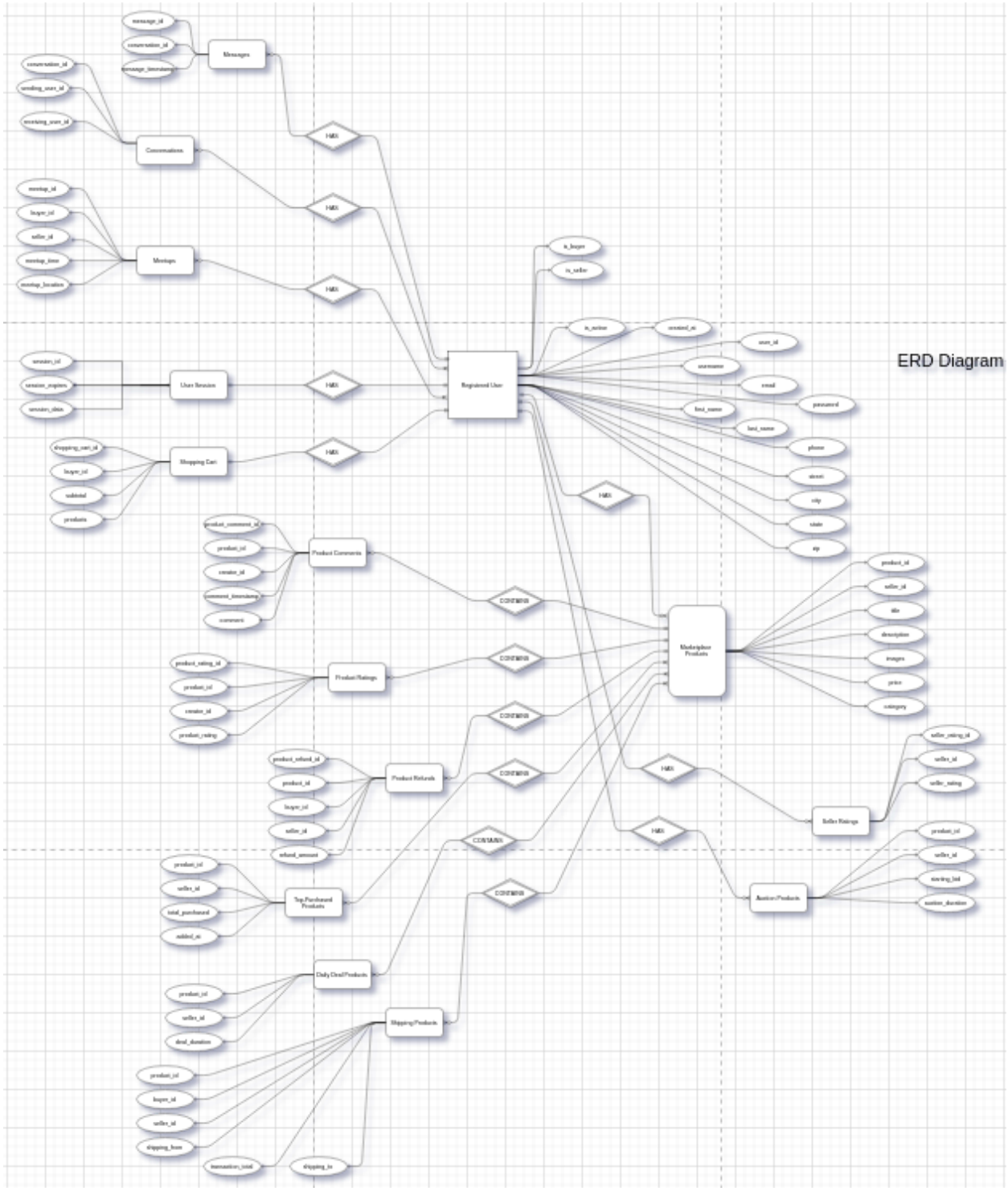
        i. users
3. Conversations
    a. Attributes
        i. conversation_id
        ii. sending_user_id
        iii. receiving_user_id
    b. Relationships
        i. users
        ii. messages
4. Messages
    a. Attributes
        i. message_id
        ii. conversation_id
        iii. message_timestamp
    b. Relationships
        i. Conversations
5. Meetups
    a. Attributes
        i. meetup_id
        ii. buyer_id
        iii. seller_id
        iv. meetup_time
        v. meetup_location
    b. Relationships
        i. users
6. Shopping cart
    a. Attributes
        i. shopping_cart_id
        ii. buyer_id
        iii. subtotal
    b. Relationships
        i. user
7. Marketplace Products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. title
        iv. description
        v. price
        vi. images
        vii. category
    b. Relationships
        i. users
        ii. product_comments
        iii. product_ratings
        iv. product_refunds
        v. top_purchased_products
        vi. daily_deal_products
        vii. shipping_products
8. Product comments
    a. Attributes

        i. product_comment_id
        ii. creator_id
        iii. product_id
        iv. comment_timestamp
        v. comment
    b. Relationships:
        i. products
        ii. users
9. Product ratings
    a. Attributes
        i. product_rating_id
        ii. product_id
        iii. creator_id
        iv. product_rating
    b. Relationships
        i. products
        ii. users
10. Product refunds
    a. Attributes
        i. product_refund_id
        ii. product_id
        iii. buyer_id
        iv. seller_id
        v. refund_amount
    b. Relationships
        i. products
        ii. users
11. Auction products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. starting_bid
        iv. auction_duration
    b. Relationships
        i. products
        ii. users
12. Top-purchased products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. total_purchased
        iv. added_at
    b. Relationships
        i. products
        ii. users
13. Daily deal products
    a. Attributes
        i. product_id
        ii. seller_id
        iii. deal_duration
    b. Relationships

        i. products
        ii. users
14. Shipping products
    a. Attributes
        i. product_id
        ii. buyer_id
        iii. seller_id
        iv. shipping_from
        v. shipping_to
        vi. transaction_total
    b. Relationships
        i. products
        ii. users

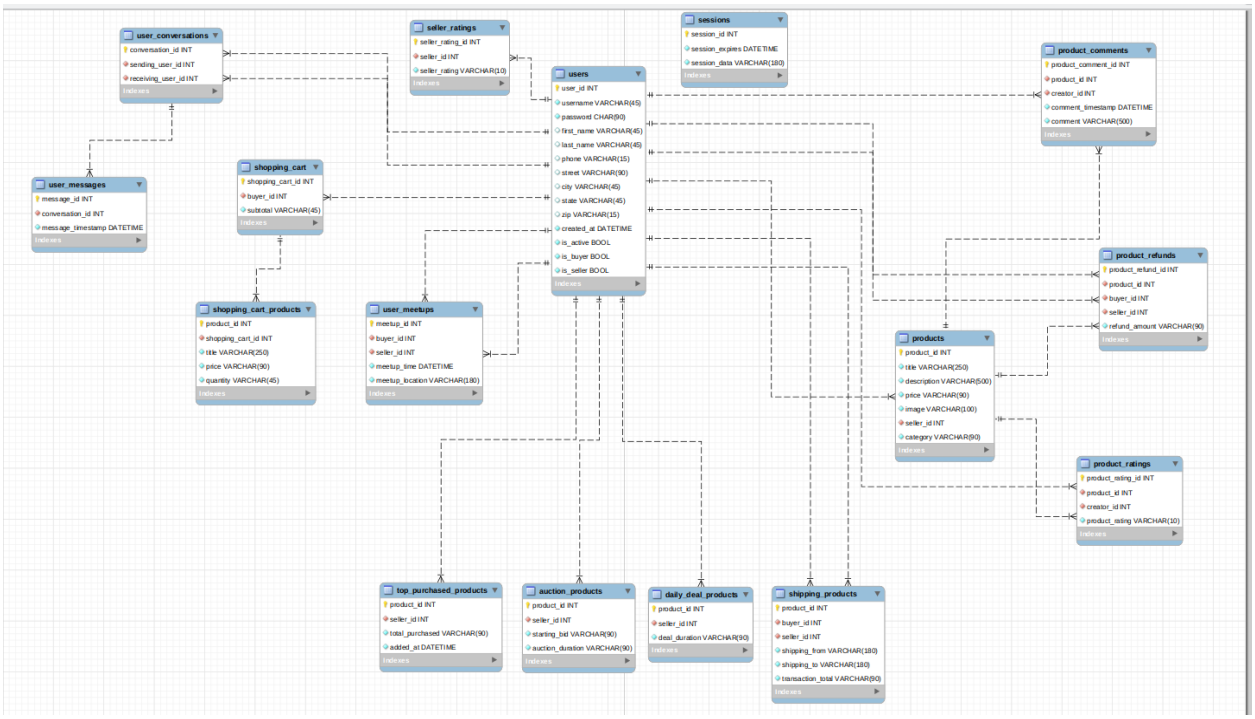
Entity Relationship Diagram

1. https://drive.google.com/file/d/1OmkYbqwzamnWdM2J0xKMSGNLJ2dQBWrc/view?usp=sharing (top of document)



Database Model

1. https://github.com/sfsu-joseo/csc648-848-sw-engineering-sum21-T03/blob/development/application/server/db/mysql_ja_model.mwb



DBMS Decision

1. We will be using MySQL Workbench as our DBMS, since the software comes out of the box with features such as writing custom SQL statements, and creating/managing models and schemas. MySQL Workbench is also an easy-to-use interface which makes creating new tables, setting public/foreign key relationships, and creating new columns simple to explain to other team members.

Media Storage

1. Images and video/audio files will be stored in our project's file system. We will only be storing the image and video/audio file names in our database in order to save space. Additionally, we have decided to store these media files in our project's file system because we can keep them in one consistent location. As a result, this makes lookup times and references to these media files easy and efficient. Since we are storing only the media file names in the database, we can simply prepend the proper path behind the image when we have to display these media files to the client. A good example of prepending the proper path would be with an <img/> tag: src={`/uploads/${product.image}`}. When we load the src attribute, we can add /uploads to the front of the file name and we're done.

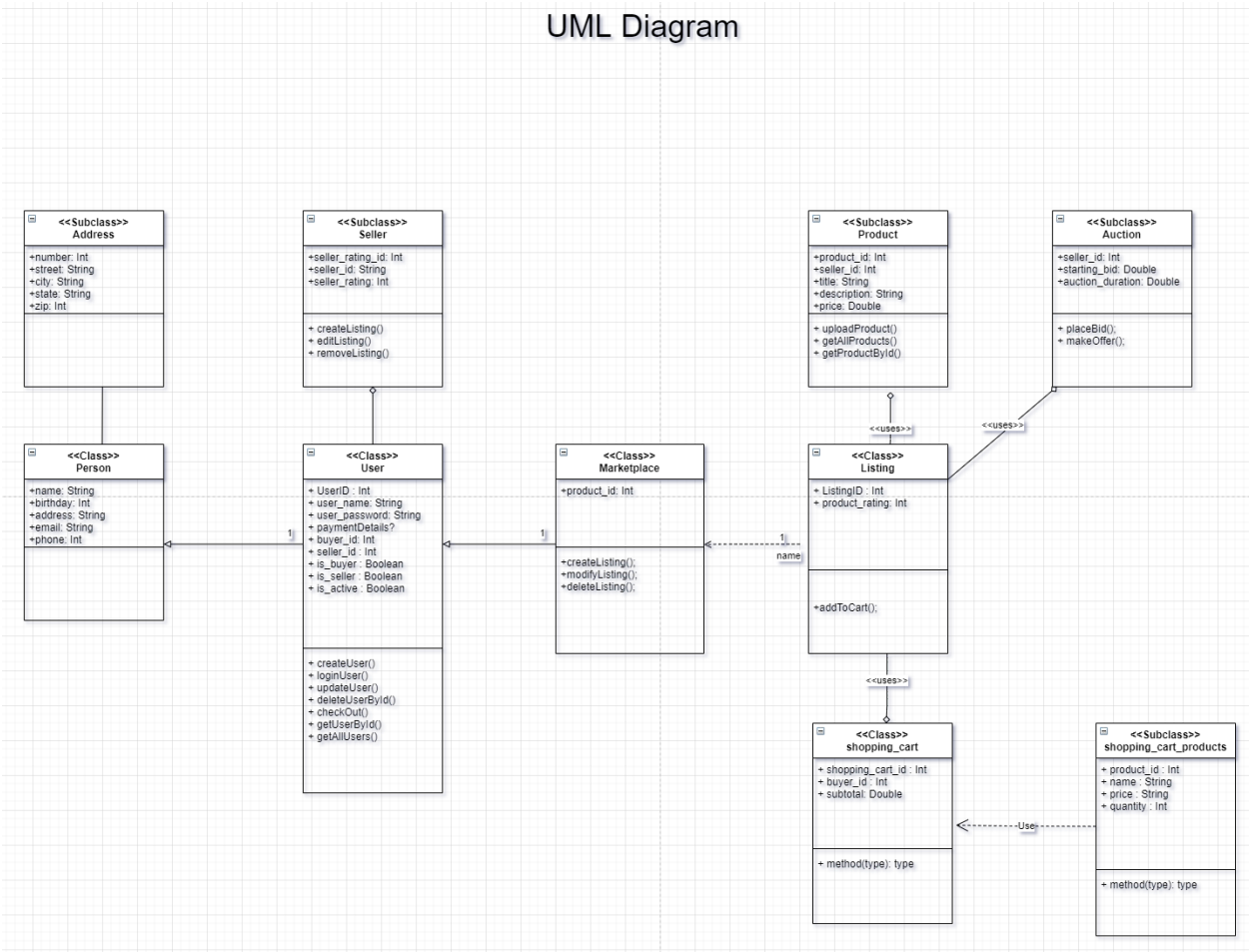
Search/filter architecture and implementation

1. Our search algorithm involves the use of a URLSearchParams object which is created when the user clicks the ‘search’ button of the search bar. We have implemented a filterProducts(products, query) function, which takes the current Array of products and the search query parameter to filter with. filterProducts() then returns an updated Array with the products corresponding to the specified search parameter. By default, our Home page will load all of the products saved in the database. This is done with a ‘SELECT * FROM products’ SQL query.
2. In order to implement the category filtering for our search bar, we have implemented an initial array of categories to choose from: Clothes, Shoes, and Electronics. When a user opens the category dropdown menu, they can choose from these options to filter products. When they click one of the category options, an axios request is sent to our Node API which takes the category they clicked as a query parameter. On the backend, the ‘/api/product-categories’ route will fetch all products from the database which correspond to the category the user clicked on. This is done with a ‘SELECT * FROM products WHERE category = ?” statement, where ? is filled in with the specified category.
3. Products will be created dynamically inside our ProductCreationForm.js component. With this component, users can specify the title, description, price, category, and image of their new product. After a user creates a new product, the product will be displayed on the Home page along with any other products.

5. High Level Diagrams

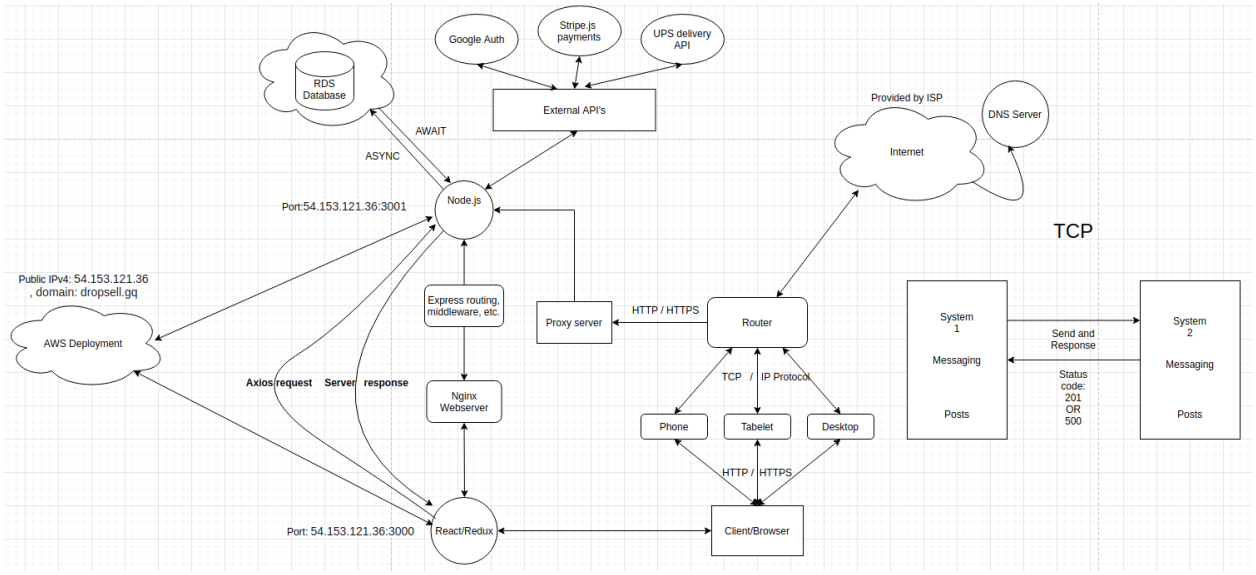
UML Diagram

1. https://drive.google.com/file/d/1OmkYbqwzamnWdM2J0xKMSGNLJ2dQBWrc/view?usp=sharing (Bottom of document)



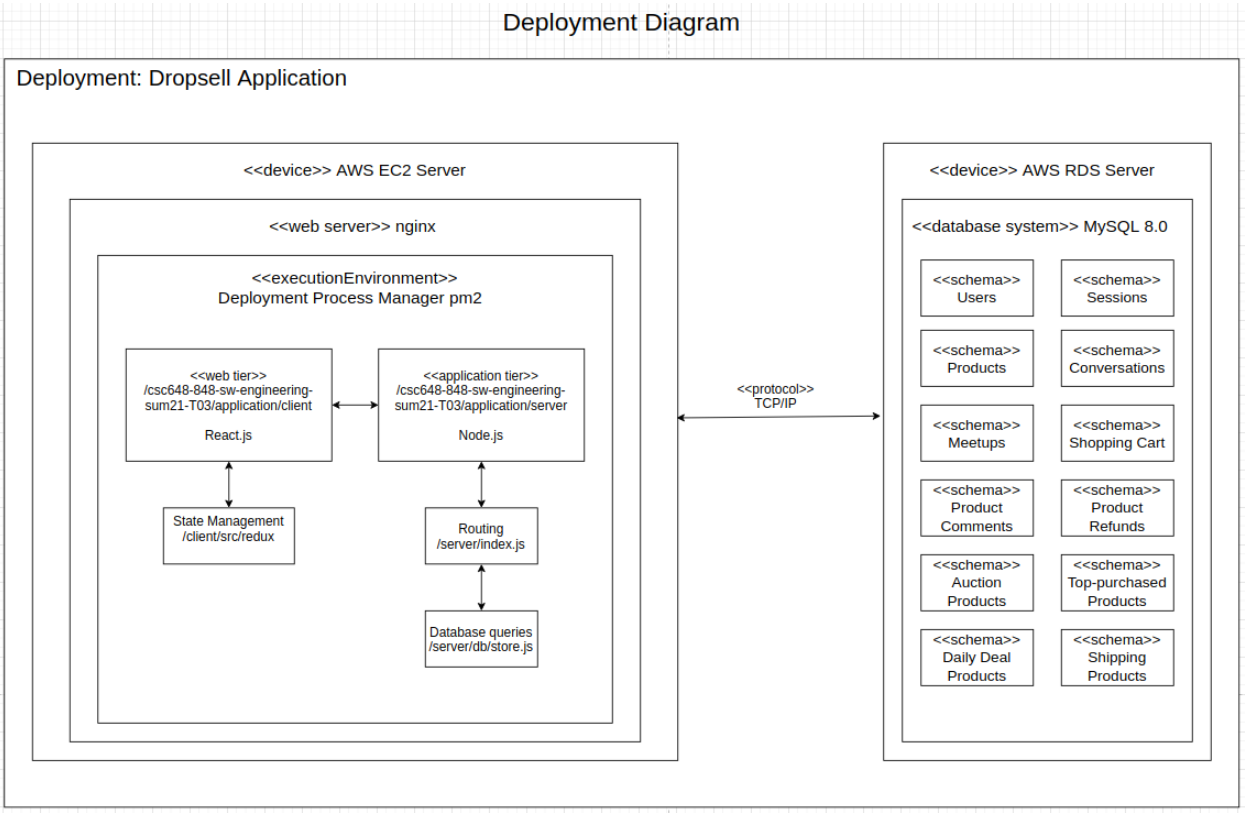
Application Network Diagram

1. https://drive.google.com/file/d/1XLbonkJqFnO7ZeOMQqstFPeZm8cr7UxX/view?usp=sharing (Top of document)



Deployment diagram

1. https://drive.google.com/file/d/1XLbonkJqFnO7ZeOMQqstFPeZm8cr7UxX/view?usp=sharing (Bottom of document



6. List of Contributions

Student name	Contributions
Mitchel Baker	Mitchel started off Milestone 3 by setting up the redux actions/reducers for register, login, and products components. He also set up the backend routes for these components, which fetched specific data and updated these values in the database for future use. Mitchel also worked with Kenneth to create the buyer and seller settings pages. Mitchel also implemented Dropsell's checkout experience pages: receipt info, summary, checkout, and final invoice, in addition to creating dynamic routes after clicking on products in the Home page. Lastly, he contributed to overall styling of Dropsell while also deploying our codebase to our AWS EC2 instance.
Charmaine Eusebio	Charmaine helped tremendously by researching the best wireframe software tools to use. She communicated what worked best for her, which was the website whimsical.com. As a result of her research efforts, she ended up creating clean wireframes which expressed exactly how we want our UI/UX to look. Charmaine also collaborated with Rowena to effectively split up the wireframes to do, which demonstrates initiative behind the tasks for the Milestone they were assigned to accomplish.
Kenneth Chuson	Kenneth played a pivotal role in the creation of our buyer and seller settings pages. He took initiative by adding libraries for seller scheduling while also jumpstarting our efforts towards implementing user analytics with the

	graph library he added. There were a few initial bugs with the buyer and seller settings pages, but Kenneth and Mitchel worked together on this to solve the problem. Kenneth implemented many of our application's new components, ranging from Activities, Account, Profile, Shipping, and WorkSchedule. Kenneth also implemented the redux actions/reducers for the buyer and settings pages. Kenneth also added the seller and buyer settings information into our data definitions section.
Krina Patel	Krina's efforts were key toward refining our Home page features. She started with a revamp of the styling we had, and then moved towards updating our NavBar component to include the sliding hamburger menu. Krina performed research on the best react UI libraries to use in regards to our NavBar component, while also implementing dynamic react components which open/close on click. After completing the NavBar, Krina also implemented our additional search filters, ranging from location, price, shipping, to condition of products.
Michael Schroeder	Michael started off Milestone 3 by updating our register UI, he also fixed the register actions for when users insert data into form inputs. As github master, Michael also helped to manage commits from branches by fixing merges and creating pull requests for features being created. Whenever there were merge conflicts, Michael communicated them to Mitchel where they were then able to solve the conflict efficiently with no time wasted. Michael has also taken initiative with our chat functionality in the application. He researched two different libraries for doing so which were cometchat and socket.io.

Rowena Echevarria	Rowena contributed to our register functionality by adding the Driver's license number input, in addition to adding the corresponding redux actions/reducer functions for this input. Rowena was also assigned to do the wireframing for our application, where she effectively split up tasks with Charmaine in order to get the job done. Rowena was at every team meeting, and always communicated the status of her progress through our Discord channels.
Jamie Walker	Jamie took initiative for Milestone 3 by updating our users schema in the database with additional information such as the user's birthdate. Jamie has also been working on implementing the Stripe API for our user checkout experience. He has been conducting a ton of research on best practices in regards to securely charging users for products they purchase. Jamie has also shared all of the information he's found in our Discord channels while also communicating to us his findings.

M3 Horizontal Prototype Feedback

1. Adjust contrast on text or add highlights
2. Add “no result” or “recommendations” on the search function
3. Add logo and move the search bar on the top
4. For registration, create a link for the terms of services instead of displaying the whole agreement form
5. Filter options could have used background colors to make them pop more
6. Contact page should be implemented
7. Provide help or customer support or something similar to FAQ’s

M4

SW Engineering CSC 648-848 Summer
2021 dropsell.gq, Jose's Angels

Team 03

Name	Email	Roles
Mitchel Baker	mbaker3@mail.sfsu.edu	Team lead
Krina Bharatbhai Patel	kpatel11@sfsu.edu	Frontend lead
Charmaine Eusebio	ceusebio1@mail.sfsu.edu	Frontend engineer
Rowena Elaine Echevarria	rechevarria@mail.sfsu.edu	Frontend engineer
Michael Schroeder	mschroeder@mail.sfsu.edu	Backend lead, Github master
Kenneth N Chuson	kchuson@mail.sfsu.edu	Backend engineer
Jamie Dominic Walker	jwalker5@mail.sfsu.edu	Backend engineer

Milestone 4
July 30, 2021

History Table

Version	Date	Notes
M4V1	07/30/2021	
M3V1	07/22/2021	
M2V2	07/20/2021	
M2V1	07/08/2021	
M1V2	07/02/2021	
M1V1	06/22/2021	

1. Product Summary

- Name of the product: Dropsell

- URL to the product accessible on deployment server: http://dropsell.gq

- How we plan to market and sell our product is two fold. First we will need some online advertisements to spread the name of our product. The top two methods of spreading our name brand online are pay-per-click (PPC), and paid social. PPC is one of the top methods for advertising your brand or product, but this involves competing with other big names for key search words. While this approach will work long term once we have established our name, it is not best for us at first due to the smaller purse and the fact that we are still trying to recuperate start-up costs. The second strategy of paid social is much more aligned with our goals. How paid social ads work is creating ads for social media platforms. This method allows us to target certain audiences that we feel our product applies to and since social media is one of the common pastimes of adults and teens this ensures we reach the most amount of people.

- The second method to build a strong user base is word of mouth. Once we get client traffic on the website we are confident that users will want to come back again and again. As users recognize the simplicity and value of Dropsell they will quickly recommend our page to their friends and family.

- What is unique about our web application drop-sell, is that we are developed from the ground up to be user friendly. Most if not all big time competitors simply added selling functionality onto their existing website. Dropsell is designed for the modern age for usability and comfort. What our web application does differently than others is that we provide tools to take the stress out of buying or selling a product. For example if you are unsure how much to charge for an item you can use our built in tool to quickly and accurately compare your product to what has sold previously or what is currently listed for sale. This takes the stress of guessing a price and underselling the product.

Itemized list of functions

1. Users for this application will be able to search for products.
2. A person that wishes to sell an item can choose to list that item directly to the market place or post it as an auction.
3. A user will be able to purchase an item by auction or buy it now style options, whichever the seller enables.
4. A user will be able to sort and filter search results to best suit their needs or interests.
5. A user will be able to see details related to their purchases in their account information.
6. If a user opted for delivery consignment upon purchase of an item, then they will be able to see tracking information in their account.
7. Instead of face to face deliver options, users can elect to have items delivered via consignment options(AKA shipping via ups for example)
8. A user shall be able to use website tools to price match items they wish to sell or purchase.
9. Any user must agree to the website's terms and conditions.
10. A user can message sellers with questions related to their products.
11. Users can add items to their shopping carts.
12. A user's shopping cart will update as the user adds or removes items from the cart.
13. A user shall be able to return back to shopping to further fill up their cart if they are midway through the check out process.
14. A user can choose to cancel their order or modify it appropriately.
15. A user can have details related to the purchase sent to their accounts for review at a later date.
16. When a user is checking out they can choose to add any additional information they want
17. When a user checks out they can choose different forms of payments.
18. A user shall receive a receipt/invoice after purchasing their item that will include

all information required(ie, purchase price, name of seller, contact info, etc).

19. Users shall agree to terms and conditions prior to be granted access to
20. Each user shall agree to a seller's fee that will be deducted once that item sells.
21. A user will be able list products for sale and will list any pertinent information.
22. A user shall be able to edit a post that they made if they deem it appropriate.
23. If a user wishes to sell more of a certain item they will be able to adjust the quantity on their post.
24. A user shall be able to delete or remove an item if they no longer wish to sell it.
25. A user will be able to check all items they have for sale in their profile/account page.
26. A user will have different options for selling products(IE a user can choose the starting bid for an item, a user can choose if that item should last for 1-3 days, etc).
27. A user can set an item’s duration on the website for a minimum of 1 hour, to a max of 30 days.
28. A user will be able to see pertinent information related to an item they are interested in purchasing(IE a user can see the current bid on an item, a user can see time remaining on an item, etc.).
29. A user shall be able to bid on an item with a single click.
30. A user shall see an accurate count down of time remaining on an item's listing time.
31. Users shall be able to contact other users

2. Usability Test Plans

Creating a new product

- Test objectives
    - For this usability test plan we will focus on adding a product to the Dropsell web application for sale. This type of sale will cover simple purchasing from another user for the listed price. Auction type sales will not be included in this test, but will be included in its own test later. Testing this feature is important because besides auction type sales this is the only way for users to purchase items, and if you can't list an item for sale, then no one can purchase it.
    - Like the auction sale type mentioned later, being able to list products for purchase is the main reason for this application. Testing this feature will allow us to learn more about the user's experience in the most critical feature of this web application and the more we learn about the user experience the more we can shape it to be the best experience possible.

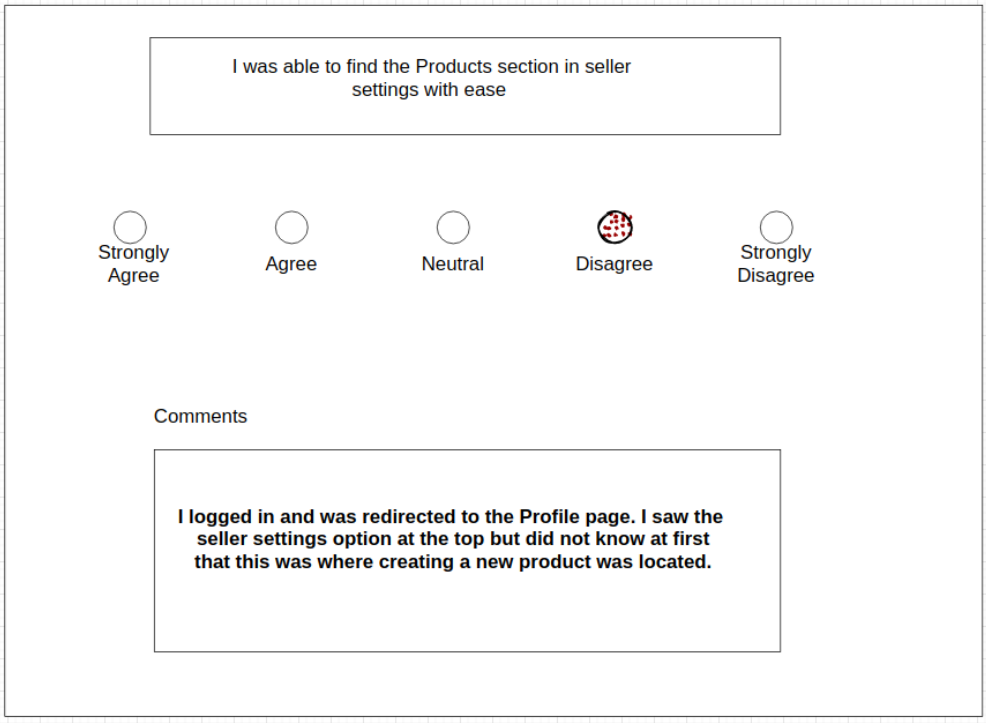
- Test description
    - The system was set up using a Linux Ubuntu system. I used multiple browsers, ranging from Brave browser, Firefox, to Google Chrome. When testing between browsers, there were no conflicts here when testing the creation of a new product.
    - The best starting point for testing the creation of a new product would be after logging in. Once the user is redirected to their Profile page, they should then click on seller settings which is where they will find the “Products” button which will display the component for creating a new product.

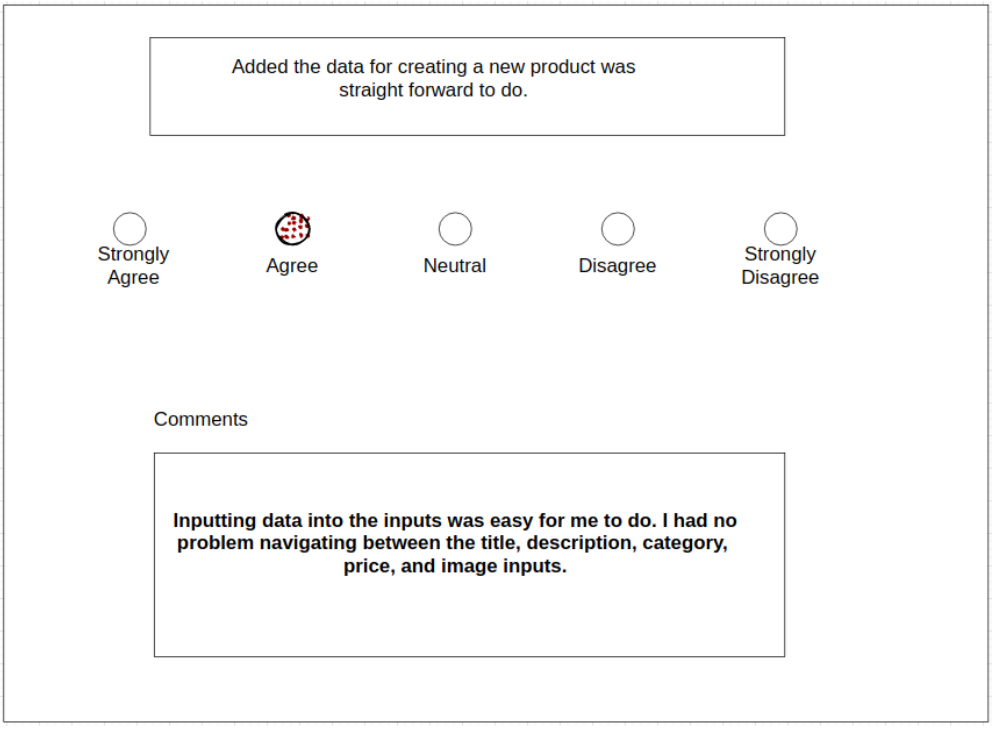
- Usability Task description
    - When thinking about the intended user base who would gravitate towards creating new products, people who are looking to clear out their garage, or sell old clothing for some extra cash come to mind. I had my girlfriend use the site to create a new product since she likes to post her used clothes online for people to purchase. She was a good candidate choice because she points out areas of the project which are lacking in UI/UX expertise. She is also a marketing major so she was able to point out details in the application related to advertising towards sellers.
- URL of the System to be Tested
    - The URL of the system which is used for testing the creation of a new product is on http://dropsell.gq/seller-settings, since the creation of a new product is located in the seller settings.

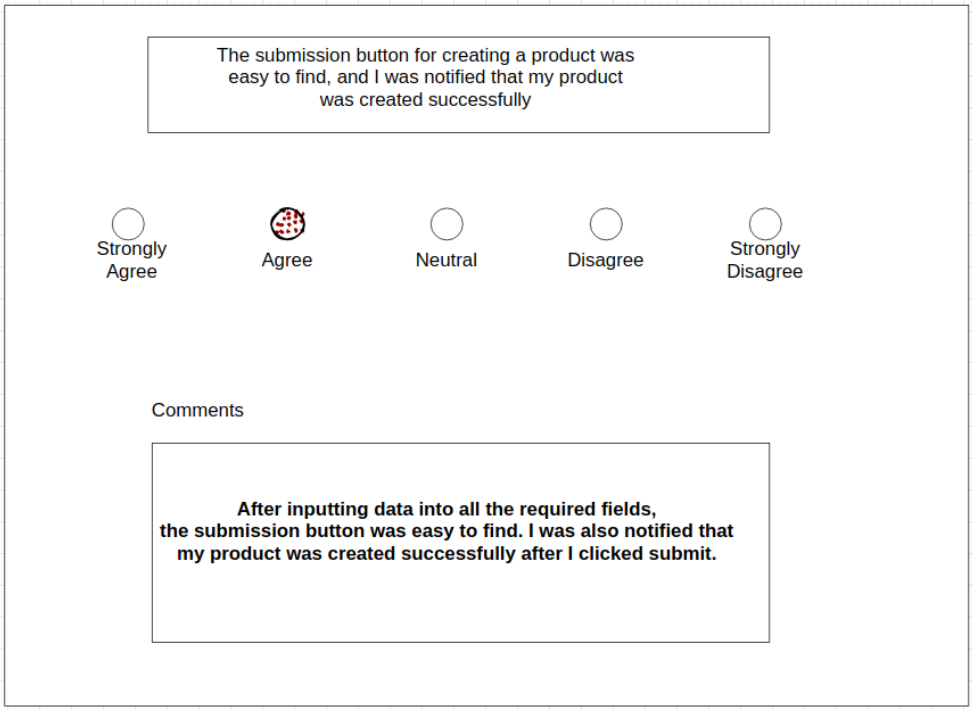
- Usability Test Table

Test/Use Case	% Completed	Errors	Comments
Locating Products in Seller Settings	80%	No errors found	Locating the "Products" in seller settings was difficult. These settings were too tucked awa
Inputting product data into the form	100%	N/A	Inputting data for the product title, description, category, price, and image were straight forward and intuitive to do.
Form validation when inputting invalid data	0%	Our nonfunctional requirements specify that product titles should not exceed 80 characters in length, while product descriptions should not exceed 500 characters in length. When testing for these system constraints, the application did not notify the user that the data inputted was incorrect.	N/A
Submit product	100%	N/A	Submitting a new product was simple. The submission button was easy to find, and I was notified when the product was successfully added.

- Questionnaires






The submission button for creating a product was easy to find, and I was notified that my product was created successfully
Strongly Agree
Agree
Neutral
Disagree
Strongly Disagree
Comments
After inputting data into all the required fields, the submission button was easy to find. I was also notified that my product was created successfully after I clicked submit.

Creating an Auction Product

- Test Objectives
    - For this test plan, the feature in question is the process of creating an auction on the site. An auction differs from a regular listing, in that it allows potential buyers to submit a price point and compete with other buyers until the time limit expires. This is beneficial to a buyer, due to potential cost savings over a regular listing. This feature is important to test, as it can be one of the main selling points of the site to users. We want to be able to test the frontend and backend features of the site to determine that the auction listing is being created, with all of the necessary information being added to the mysql database.
    - Understanding the functionality of creating this auction is very important to understanding how the company can innovate going forward. There are other features of the site that allow us to pick up items from customers to sell ourselves, to alleviate the stress and frustration of selling items online. Allowing this system to flourish under these testing conditions, then we can assess the impact that this will have on our growth as a company.

- Test Description
    - This system was set up using the Windows 10 operating system. I used several different browsers throughout this test, with minimal conflicts. I tested this feature using Chrome, Firefox, and Microsoft Edge. There were zero issues with these browsers and operating system combination.
    - A good starting point for this testing environment would be right after login, at profile settings. We want to be able to switch over to the seller-settings page from the profile page after logging in. Once we are there, we can determine what kind of listing we want to have. Is this an instant purchase listing, or an auction style listing?

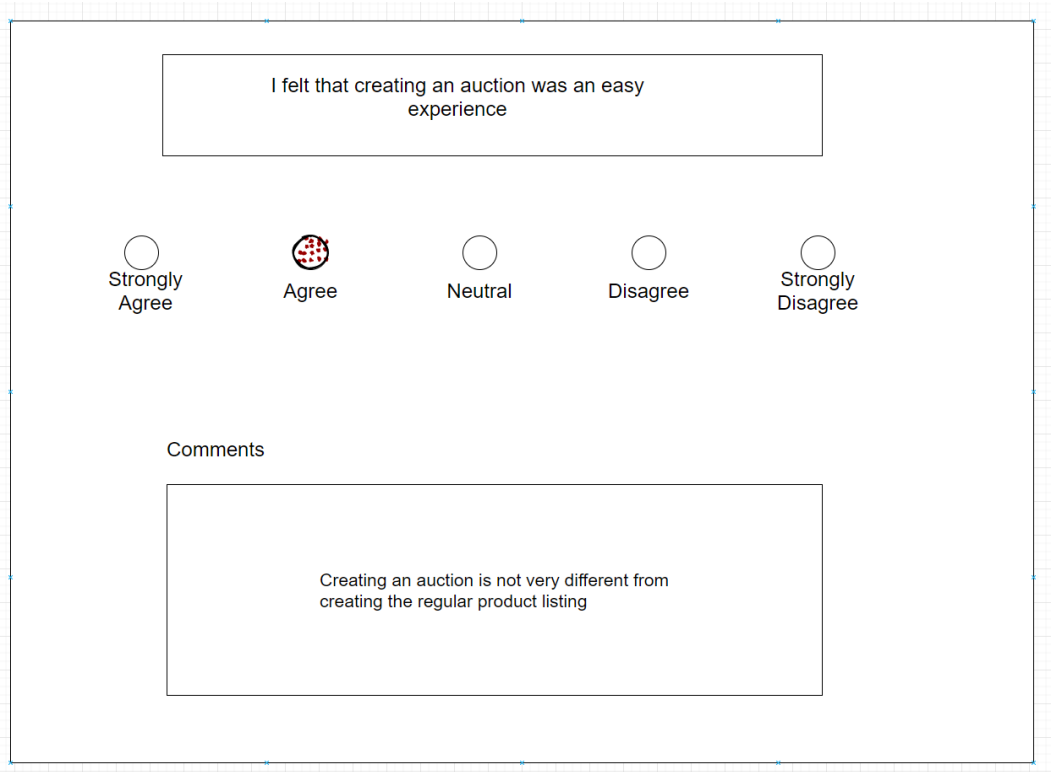
- Usability Task Description
    - The intended users of the platform are anyone looking to lighten the load or do some sort of spring cleaning. The intended users for the testing environment are the same folks. I used my wife (who is not very technically savvy) to try and attempt creating an auction page. She was a perfect candidate as she is a professional at constantly finding errors or flaws in my designs. I believe she represents what most people look for in a testing candidate, someone to come in and try and break what you think works fine.

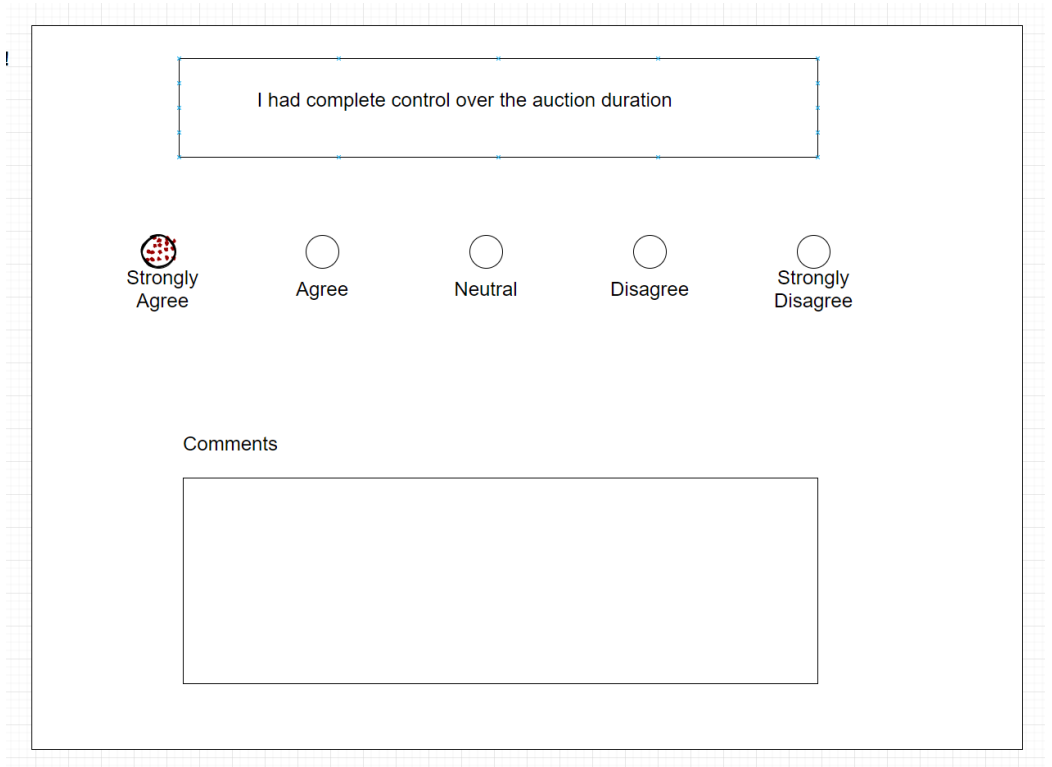
- URL of the System to be Tested
    - The URL of the system to be tested will be at http://dropsell.gq/seller-settings , the creating a listing in auction format is the feature to be measured. We are also testing for specific parts of the auction for usability, such as the duration, the length, and the ability to place bids and outbid other users.

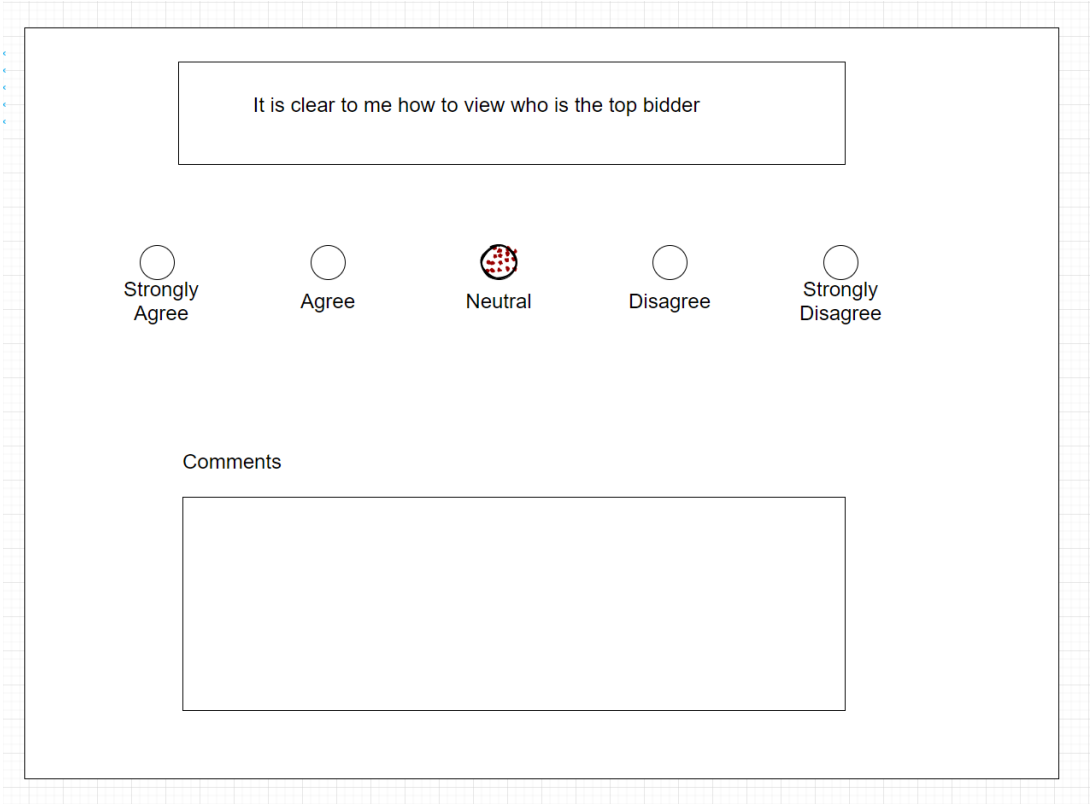
- Usability test table

Test/Use Case	% Completed	Errors	Comments
Create product auction	100%	No errors in creating a product listing	Creates listing but without all features
Set duration	75%	Specified minute duration not available	Exact start date and end date are implemented
Set minimum bid	95%	Bidding feature not implemented	N/A
Auction is found under search	50%	Specific auction listings are not available yet	N/A

- Questionnaires






It is clear to me how to view who is the top bidder
Strongly Agree
Agree
Neutral
Disagree
Strongly Disagree
Comments

Adding products to cart / Checkout process

- Test objectives
    - For this testing process, there are two functional requirements being tested here. First, we want to test the ability to add to the user's shopping cart. The second functional requirement involves completing the checkout process of said item. We want our test to be able to acknowledge that an item can be added to the cart, and remain in the cart for several different webpages. When the user is ready, the checkout process can begin. This test should acknowledge that the product's persistence remains through the cart, to the completion of the transaction. Once the transaction is completed, the item is then removed from the cart, and from the rest of the website as well.
    - Understanding the functionality of creating this auction is very important to understanding how the company can innovate going forward. Being able to have a fluid cart and checkout process is the backbone of our company's entire product. Allowing this system to flourish under these testing conditions, then we can assess the impact that this will have on our growth as a company, as well as determine the stability and usability for our testing group, or the average user.

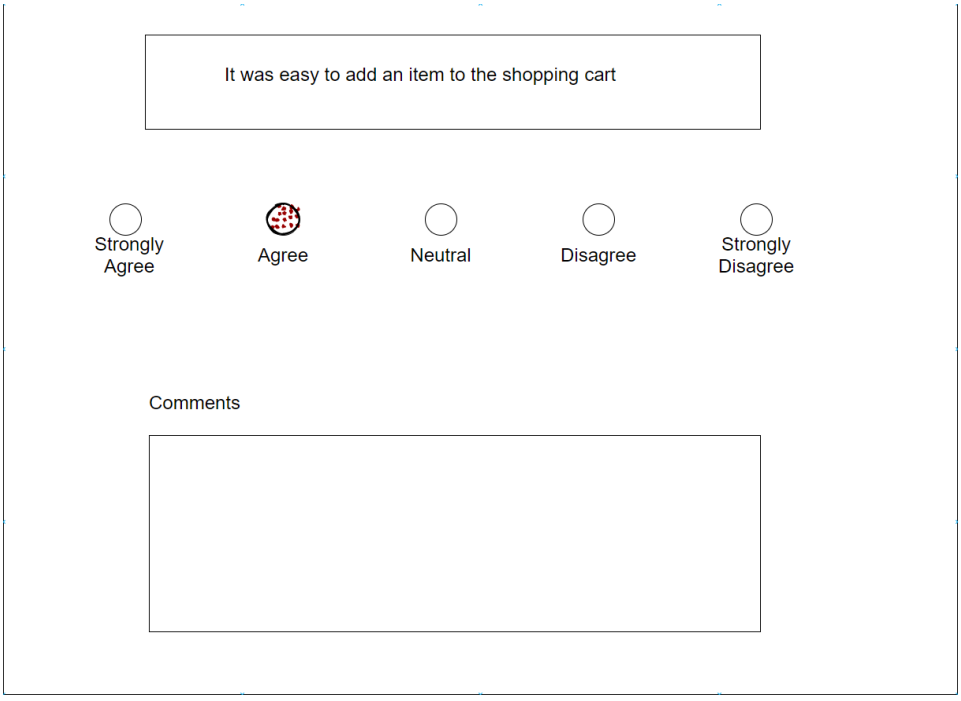
- Test description
    - This system was set up using the Windows 10 operating system. I used several different browsers throughout this test, with minimal conflicts. I tested this feature using Chrome, Firefox, and Microsoft Edge. There were zero issues with these browsers and operating system combination.
    - I decided that the best starting point is the home page. Let the user add the item to the cart, move around different web pages to signify the item is still within the cart, and then begin the checkout process whenever they are ready. Once that process had started, I wanted to be able to test that the entire checkout functionality was tested. Entering payment information, reading over the invoice, and hitting submit order. All of these may seem minor to the user, but being able to test each one individually is crucial to the usability of our product.
    - The intended users of this product are the same as the intended users from the usability testing of both creating the listing, and creating an auction. These are people who want to purchase an item that they are selling, whether it is to finally grab that Xbox that is sold out elsewhere, or to bring to fruition the idea of, another man's trash is another man's treasure.

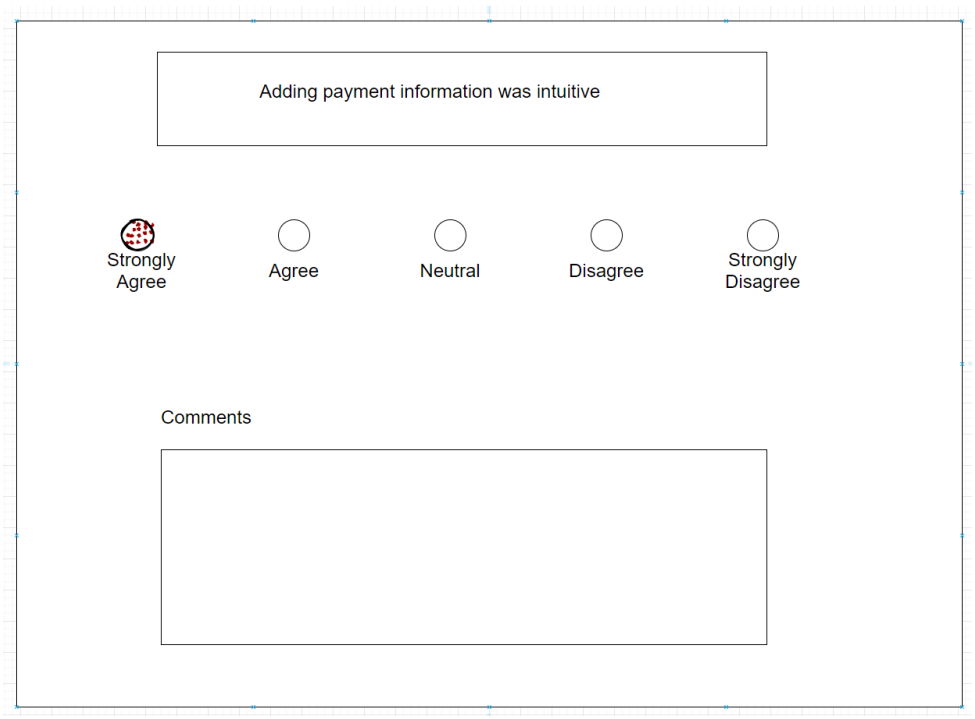
- URL of the system to be tested and what is to be measured
    - The URL of the system to be tested will just be the home page as a good starting point. (http://dropsell.gq). The system that is to be tested and measured, will be the checkout process. Adding an item to the cart will not change the URL, unless the site is redirected to a shopping cart screen. Once on this screen, the user can be given the option of completing checkout. If the customer proceeds with completing checkout then the process of entering payment info is to be tested. Persistence of payment information and items added to cart are to be tested by the user as well.

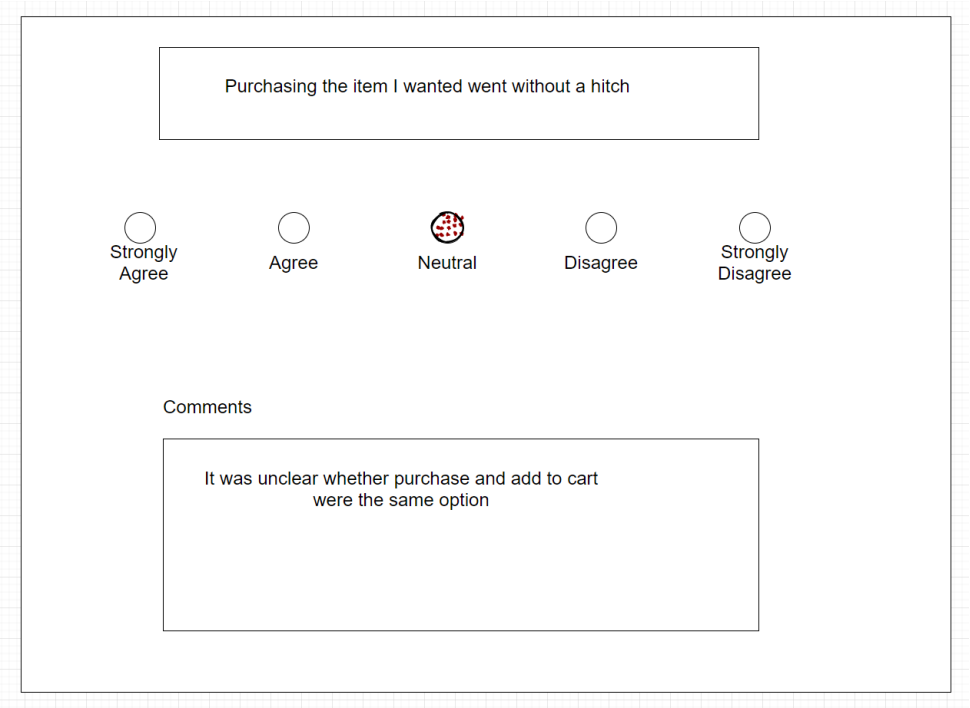
- Usability test table

Test/Use Case	% Completed	Errors	Comments
Persistent shopping cart	100%	Shopping cart remains full across pages	This done by adding product to the mysql database
Purchase product before opening shopping cart	75%	Application will error out if product is not purchased first before opening up the shopping cart from the hamburger menu	Workaround to this issue to click purchase button to add into the cart
Add payment information	90%	Payment information stored in checkout process, but not persistent in account management	N/A
Submit order	100%	N/A	N/A

- Questionnaires


It was easy to add an item to the shopping cart
Strongly Agree
Agree
Neutral
Disagree
Strongly Disagree
Comments


Adding payment information was intuitive
Strongly Agree
Agree
Neutral
Disagree
Strongly Disagree
Comments


Purchasing the item I wanted went without a hitch
Strongly Agree
Agree
Neutral
Disagree
Strongly Disagree
Comments
It was unclear whether purchase and add to cart were the same option

Price Matching

- Test objectives
  - The objectives of this test are to determine whether the ability to price match using web scraping tools is efficient and functional. Our main objective is not only usability, ease of use, but also functionality. We want to make sure that we are able to determine the prices of competitors so that our potential users can make informed decisions, which ultimately lead them to use our site over the competition.
  - Our price matching setup currently scrapes prices from www.amazon.com, but will be added to support more websites as well. We want our tests to show to the potential user that it is easy to use, and works as intended. This feature is not only important to the regular everyday user, but also for people looking to sell as well. Being able to see what a competitor is offering on the same product, and being able to slightly undercut them to achieve that sale, is what using our website should be all about. Having all of these features baked into the core product is very important.

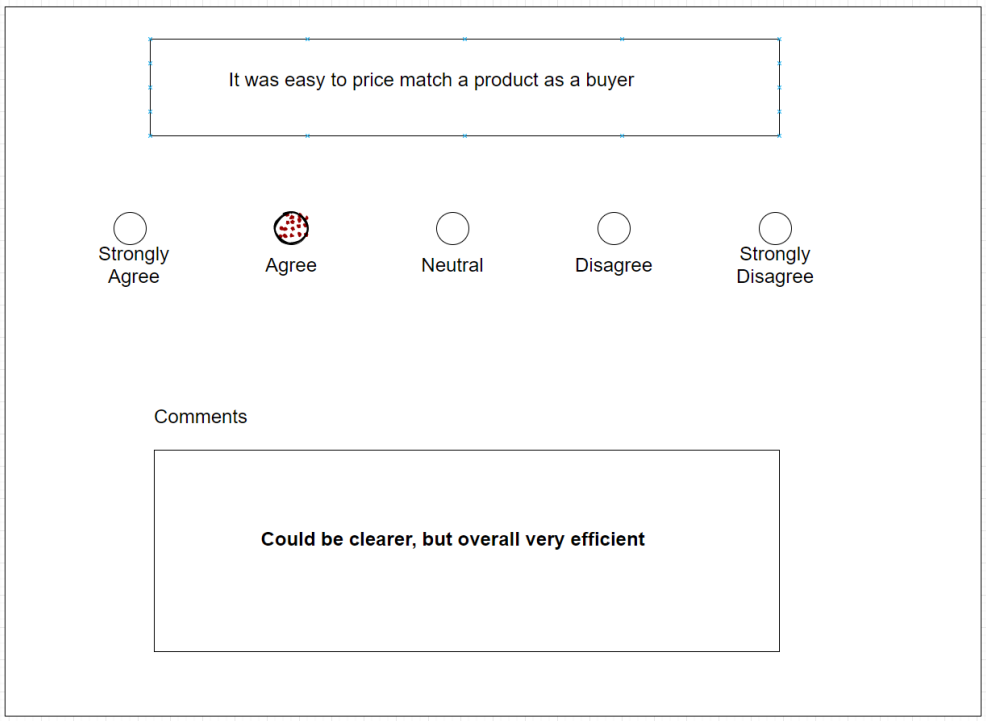
- Test description
  - There will be several tests done for price matching. The first test conducted will be to determine whether or not figuring out how to price match was user friendly. Was it an easy task to figure out how to do? Were the users able to complete a sale without even realizing it was a feature? These are all considerations being taken to understand how this will be accomplished. There will be another test to determine whether there was more than one site. We also want to find out if the design is simple enough for the average user to figure out how to price match against more than just Amazon.
  - It is very important as well for the buyer to be able to determine how price matching works on the buyer's side. Do they have access to the same functionality outside of seller settings? These are forms that will be tested, as it is very important to our core mission that we want it to be as easy as possible for both buyers and sellers to make the most informed decision that they can on this subject. Being able to determine if an item is cheaper on our website, will help secure the sale for both the seller and the buyer, at a lower rate than shopping at the big box stores and websites.

- URL of the system to be tested and what is to be measured
  - The URL of the system to be tested will be at http://dropsell.gg/seller-settings, and as well as the product listing page itself. What will be measured is the ease of use and accessibility of the price matching feature itself. Does the link to Amazon bring up the relevant product, or a similar product in price/scale? We want to make sure that the feature makes sense from a front end standpoint as well as a back end standpoint. Is it possible for it to point to more than one competitor, and how well it does the job at scraping the data from those websites to determine the accuracy of the product being searched.

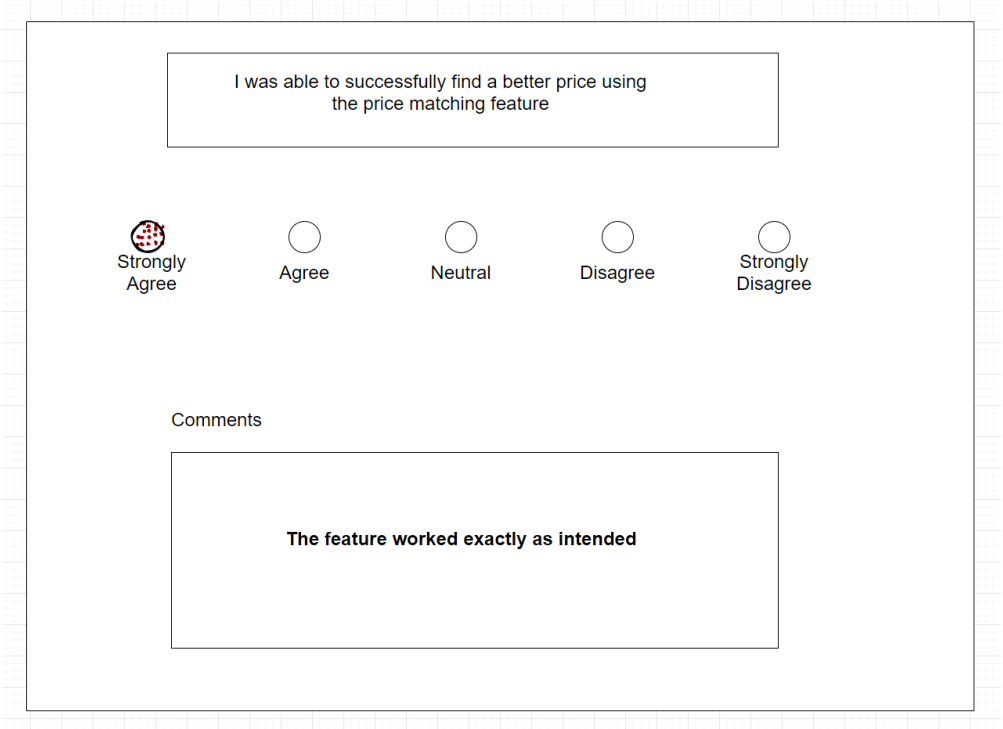
- Usability test table

Test/Use Case	% Completed	Errors	Comments
Price matches to same product	95%	N/A	Sometimes product isn't always exact
Price matches to similar product	100%	N/A	Product will always be matched to similar if not the same exact product
Price matching on more than one website	0%	Price matching only currently occurs on Amazon.com	N/A
Buyer is able to price match from product page	100%	N/A	N/A

- Questionnaires



-




I was able to price check from several different online retailers
Strongly Agree
Agree
Neutral
Disagree
Strongly Disagree
Comments
Currently I am only able to price check from Amazon

Seller/Buyer scheduling

- Test Objectives
    - The seller/buyer schedule calendar feature is to ability and organize meetup times between the buyers and sellers. This can help to improve organizing and managing plans, meetups, and work times. Sellers should be able to have this feature because it helps them to be able to track their present and future tasks. This can also improve business to view the history between sellers and buyers meetup ups and making sure to keep track if there are economic issues. Also, the schedule calendar meetup notes are able to record how the sellers set up their good meet up times performance.
    - This feature is important for business management because it can analyze buyers and sellers meetup times and records the performance for the business. Also, ability to store the meetup notes data onto the database whenever sellers and/or buyers put their meetup notes on the schedule calendar feature.

- Test Description
    - This system was set up using the Windows 10 operating system. I used several different browsers throughout this test, with minimal conflicts. I tested this feature using Chrome and Safari. There were zero issues with these browsers and operating system combination.
    - When you do not have an account for the dropsell application, you need to register an account, then login in. Then, it goes directly to your profile page. If you are a seller, you need to go to the seller settings and click “Set Up Schedule”. After that, you are able to fill up your title, start meeting time, end meeting time, and location. Once you click submit, it goes to the calendar and records your meetup note and you are able to cancel the meetup note if you want. Also, making sure that the library is working and can handle the feature component properly and handling inputs for displaying in front of the calendar schedule.
    - The difficult part of testing this feature is when a seller decides to cancel the meeting, then the buyer does not want to let the seller cancel the meeting, and there might be a time conflict issue. The important testing is whenever a seller or buyer logs out from an application, it should record and not automatically delete their meeting notes.

- URL of the system to be tested and what is to be measured
    - The URL of the system to be tested will be at http://dropsell.gq/set-workSchedule, on the title, start time, end time, and location inputs are to test up if the meeting notes are actually displayed on the calendar.

- Questionnaires

Test/Use Case	% Completed	Errors	Comments
Add Meetup Note	100%	No errors in add meetup note	Creates the list of meetup notes
Display Meetup Note	80%	Handling start and end meeting up error	Making sure the start time and end time have the correct order
Delete Meetup Note	100%	No errors in delete meetup note	N/A
Edit Meetup Note	50%	Must edit the meetup notes	Possibly can change the meetup notes, but not editing the title, start time, end time, and location.

- Testing Table

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Library	Test the calendar react library	Install and get Calendar react library	Display calendar react on the set up schedule component	Pass
2	URL	Check if the calendar react is working when hosting	Display calendar react while hosting	Set up schedule link route	Pass
3	Database	Records meeting notes	mySQL data	Prints and records the meeting notes	Fail
4	Functioning Calendar schedule	Ability to add, edit, and delete meeting notes.	User meeting notes inputs	Change meeting notes	Pass
5	Socket	Records meeting notes between the buyers and sellers	Add meeting notes and tag a buyer/seller	Should able to see the meeting notes between a buyer and a seller	Fail

2. QA Test Plan

- Test Objectives
    - Our objective with these tests is to verify that Dropsell's superior feature - price matching is handled properly or not. Due to the fact that this feature is our main focus as it is not implemented by our competitor sites, we need to ensure that this functionality is working reliably. We will be targeting specific tests such as verifying if the puppeteer library is scraping the similar product data from amazon. Additionally, we will also test our ability to store and retrieve product information on the database, which will be stored in a separate table.

- HW and SW Setup
    - For all features, the hardware needed is a computer or laptop with power, running on Windows, Ubuntu or MacOS operating system which is connected to a stable internet connection. Users will need a browser such as Chrome or Firefox installed on their computer. Last setup requirement is to navigate to a browser and go to Dropsell via http://dropsell.gq/

- Features to be Tested
    1. Library
        1.1. Test the high-level browser automation library used for Price Matching
    2. URL
        2.1. Check if amazon's url is given to create http header
    3. Parsing
        3.1. Check how the data extracted from web scraping is converted into structured format
    4. Database
        4.1. Check if the extracted data is getting stored in mysql table 'puppeteer_data'
    5. Internal Comparison
        5.1. Check if Price Matching is applied to compare price with other products of Dropsell

- QA Test Plan

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results
1	Library	Test the high-level browser automation library used for Price Matching	Product name	Prices of the same product that are for sale on Amazon	Pass
2	URL	Check the http header	Send http request to Amazon API gateway	Amazon URL	Pass
3	Parsing	Check if the scraped data is converted into structured format	Product title, image, price, seller	JSON format of Product title, image, price, seller	Pass
4	Database	Check if the extracted data is getting stored in database	JSON data extracted from web scrapping	new row in 'puppeteer _data' table	Pass
5	Internal Comparison	Check if Price Matching is applied to compare price with other products of Dropsell	Browse a Product	Price matching results with Dropsell products	Fail

3. Code Review

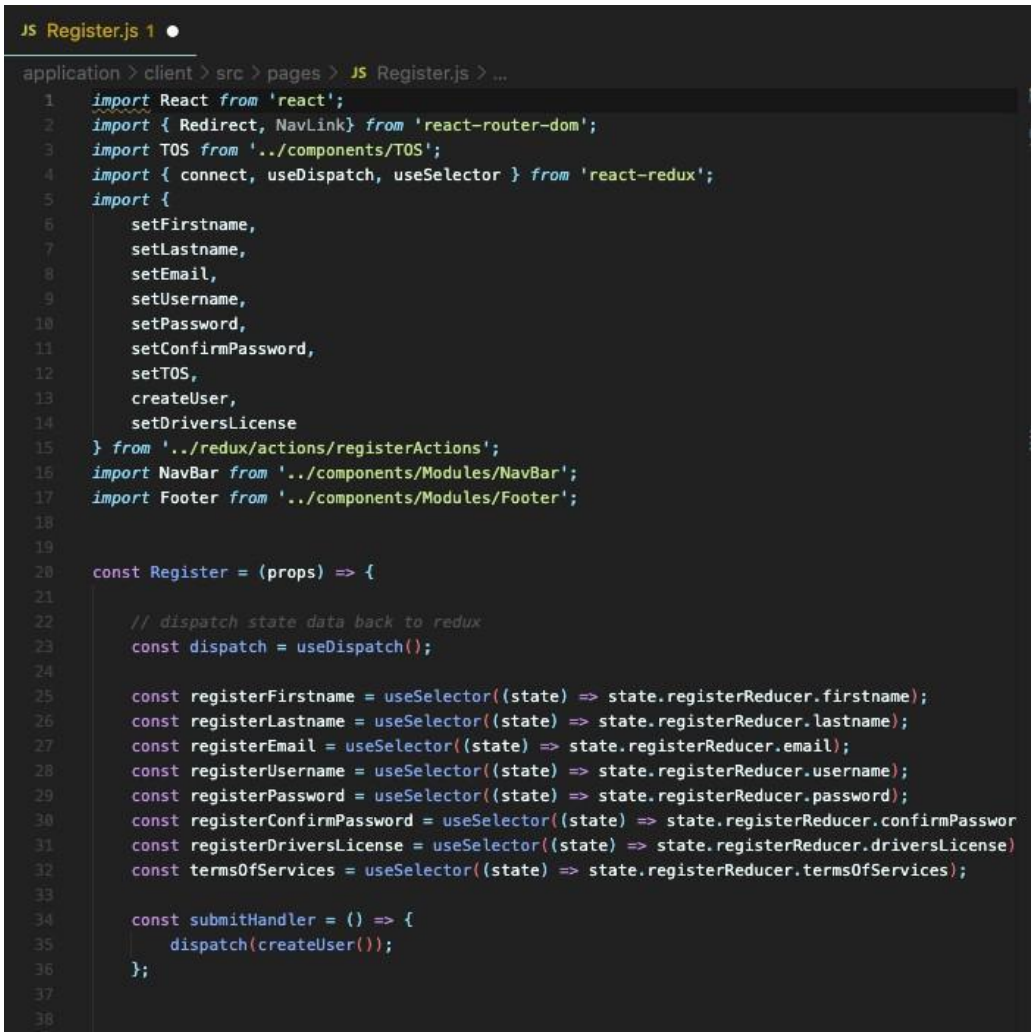
a) Coding style

i) The coding style we chose is to have separation code for the front end which is in the client directory and back end for the server directory. For the front end, we decided to have pages and the separation of components. The pages are responsible for handling components when rendering an application. The components have different features of each page such as settings, landing page, about, etc. The module's component is to have different partial features for the pages in the client side such as navigation bar, search bar, menu, and etc. So, these partial features are the children for the pages and the partial features have their own functionalities in the entire application.

ii) When it comes to handling user inputs, buttons, checkboxes, and etc. These features are where actions and reducers are responsible for keeping track of. Inside the redux directory, we have actions and reducer directories. Actions is where getting the user actions input data from the feature components. Then, using axios, is where to work with an API and be able to communicate between frontend and backend for requesting and responding for the user input data.

b) Example of our application code style

So let's start with our front end first. The client directory is where we handle most of the feature components. Let’s say we use registration of how this page works for both front end and back end as an example.



```
JS Register.js 1 ●
application > client > src > pages > JS Register.js > ...
1   import React from 'react';
2   import { Redirect, NavLink} from 'react-router-dom';
3   import TOS from '../components/TOS';
4   import { connect, useDispatch, useSelector } from 'react-redux';
5   import {
6       setFirstname,
7       setLastname,
8       setEmail,
9       setUsername,
10      setPassword,
11      setConfirmPassword,
12      setTOS,
13      createUser,
14      setDriversLicense
15  } from '../redux/actions/registerActions';
16  import NavBar from '../components/Modules/NavBar';
17  import Footer from '../components/Modules/Footer';
18
19
20  const Register = (props) => {
21
22      // dispatch state data back to redux
23      const dispatch = useDispatch();
24
25      const registerFirstname = useSelector((state) => state.registerReducer.firstname);
26      const registerLastname = useSelector((state) => state.registerReducer.lastname);
27      const registerEmail = useSelector((state) => state.registerReducer.email);
28      const registerUsername = useSelector((state) => state.registerReducer.username);
29      const registerPassword = useSelector((state) => state.registerReducer.password);
30      const registerConfirmPassword = useSelector((state) => state.registerReducer.confirmPasswor
31      const registerDriversLicense = useSelector((state) => state.registerReducer.driversLicense)
32      const termsOfServices = useSelector((state) => state.registerReducer.termsOfServices);
33
34      const submitHandler = () => {
35          dispatch(createUser());
36      };
37
38
```

We made a functional base which is called “Register” and in line 6 through 14 is where the variables that we need to use for register redux action. Also, in line 25 through 32 is where we use for redux actions and reducer that can handle for user inputs, checkboxes, and buttons.

```
<div className="register-wrapper-first-last">
    <div>
        <p>First Name</p>
        <input className="register-firstname"
                tyep="text" placeholder="First name"
                autoComplete="First Name"
                value={registerFirstname}
                onChange={(e) => dispatch(setFirstname(e.target.value))}
                required/>
    </div>
    <div>
        <p>Last Name</p>
        <input className="register-lastname"
                tyep="text" placeholder="Last name"
                autoComplete="Last Name" value={registerLastname}
                onChange={(e) => dispatch(setLastname(e.target.value))}
                required/>
    </div>
```

In the JSX, in the attribute of onChange, dispatch is coming from react hooks and it is important working with reducer then takes the action functions such as “setFirstname”, “setLastname”, and etc.

```
JS Register.js 1 ●
application > client > src > pages > JS Register.js > [@] Register
                    </div>
                    <div className="register-wrapper-confPassword">
                        <p>Confirm Password</p>
                        <input className="register-confPassword" type="password" autoComplete="new-password" placeholder="Confirm Password" value={regi
                    </div>
                    <div className="dl-wrapper">
                        <p>Only enter if signing up as a Seller</p>
                        <input className="register-dl" type="text" autoComplete="DL#" placeholder="Driver's License ID" value={registerDriversLicense}
                    </div>
                </label>

                <p> <b>Terms of Services and Privacy Agreement</b></p>
                <TOS/>

                <div className="tos-checkbox">
                    <span className="tos-checkbox-text"> By checking the box, you agree to Dropsell's Terms of Services and Privacy Agreement. </span>
                </div>

                <button className="submit-signup" onClick={submitHandler}> Register </button>
            </div>

            <Footer/>
        </div>
    );
  }
};

function mapStateToProps(state) {
    return { registered: state.registerReducer.registered };
}

export default connect(mapStateToProps)(Register);
```

We use mapStateToProps for extracting data that takes the state of the register reducer.

registerActions.js 1

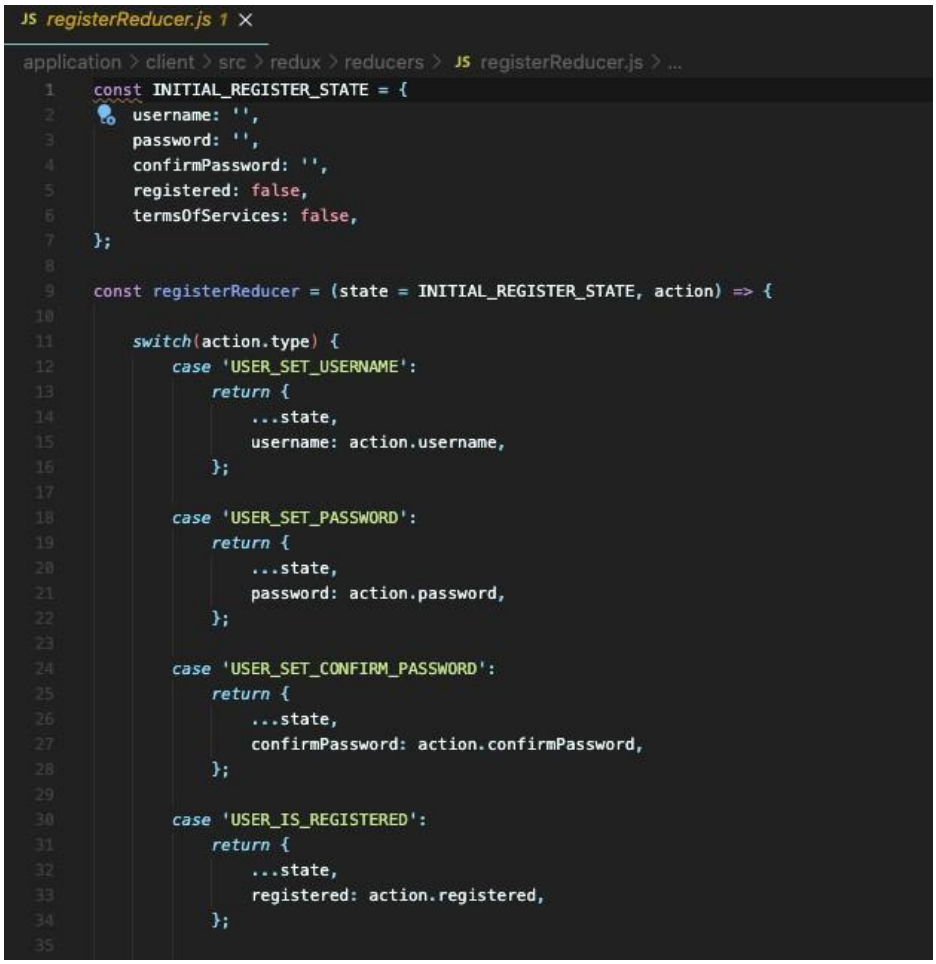
application > client > src > redux > actions > registerActions.js > setEmail

```
import axios from 'axios';

export const setFirstname = (firstname) => ({
    type: 'USER_SET_FIRSTNAME',
    firstname
});

export const setLastname = (lastname) => ({
    type: 'USER_SET_LASTNAME',
    lastname
});

export const setEmail = (email) => ({
    type: 'USER_SET_EMAIL',
    email
})

export const setUsername = (username) => ({
    type: 'USER_SET_USERNAME',
    username
});

export const setPassword = (password) => ({
    type: 'USER_SET_PASSWORD',
    password
});

export const setConfirmPassword = (confirmPassword) => ({
    type: 'USER_SET_CONFIRM_PASSWORD',
    confirmPassword
});

export const setTOS = (TOS) => ({
    type: 'USER_SET_TOS',
    TOS
});

export const setDriversLicense = (driversLicense) => ({
```

Now inside the action directory, we have the list of set action functions for handling from register inputs, checkboxes, and buttons events.

```
export const createUser = () => {
    return (dispatch, getState) => {
        const userData = {
            firstname: getState().registerReducer.firstname,
            lastname: getState().registerReducer.lastname,
            email: getState().registerReducer.email,
            username: getState().registerReducer.username,
            password: getState().registerReducer.password,
            confirmPassword: getState().registerReducer.confirmPassword,
            driversLicense: getState().registerReducer.driversLicense
        };

        console.log(userData);

        axios.post('/api/register', userData)
            .then((res) => {
                console.log(res);
                if(res.status === 201) {
                    dispatch(redirectUser(true));
                }

            })
            .catch((err) => {
                console.log(err);
            });
    };
};

export const redirectUser = (registered) => ({
    type: "USER_IS_REGISTERED",
    registered
});
```

When the user is done signing up and clicks the register button, it goes to the action createUser function and takes the firstName, lastName, and all register inputs data. Then, using axios to go to the api route and respond to the back-end server.



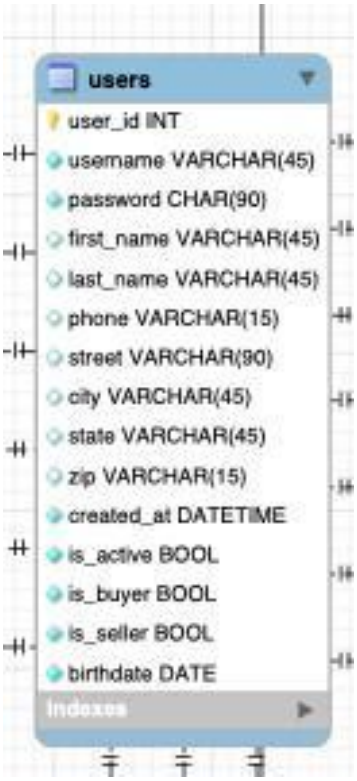
The registration reducer takes from the registration input handler events by using mapStateToProps.

```
app.post('/api/register', (req, res, next) => {
    const { username, password, confirmPassword} = req.body;

    if (username && password && confirmPassword) {
        console.log(username + " " + password + " " + confirmPassword);
        next();
    }
    else {
        res.status(400).send({
          error: "The payload is wrong!"
        });
    }
},
(req, res) => {
    store
        .createUser(req.body.username, req.body.password)
        .then((createdUser) => {
            console.log(createdUser);
            res.status(201).send(createdUser);
        })
        .catch(error => {
            console.log(error);
            res.status(500).send({ error: "Unable to insert the user" });
        });
});
```

In the server directory in index.js, this function will take or is requesting the username and password of an existing user for our application for the login feature.

```
async function createUser(username, password) {

  const encPassword = await bcrypt.hash(password, saltRounds);
  console.log(encPassword);

  const date = new Date();

  const result = await pool.query(
    "INSERT INTO users SET username = ?, password = ?, created_at = ?, is_active = ?, is_b
    [username, encPassword, date, 1, 1, 0]
  );
  if (result[0].length < 1) {
    throw new Error(
      `Failed to create a new user ${username}`
    );
  }
  return getUserById(result[0].insertId);
}
```

In the store.js, will take the mysql query that inserts the username and password to the database when creating a user from a registration.



Inside the mySQL workbench is where the users attribute data is located.

4. Self-Check on best practices for security

1. Major assets being protected
    a. Registered user account information
        i. Password
    b. AWS EC2 instance
        i. SSH URL
        ii. SSH username
        iii. SSH RSA Key
    c. RDS MySQL database
        i. Database username
        ii. Database password
        iii. Database name

When a user creates a new account, the account information they've inputted is sent securely to the `/api/register` route in our Node.js backend. The user's data is first processed in the first body of the `/api/register` route. For validating user input in the register route, we are using the validator npm library. To validate user registration data, we test to determine if the email is the correct format, if the first name/last name/username/password/confirm password are filled in or not (if these inputs are empty then they are not valid), and if the password and confirm password inputs are equal to one another. If all these tests pass for user registration, then this data is passed as parameters into the .createUser() function written in store.js.

```
app.post('/api/register', (req, res, next) => {
    const { firstname, lastname, email, username, password, confirmPassword, driverLicense} = req.body;

    if (validator.isEmail(email) &&
        !(validator.isEmpty(firstname)) &&
        !(validator.isEmpty(lastname)) &&
        !(validator.isEmpty(username)) &&
        !(validator.isEmpty(password)) &&
        !(validator.isEmpty(confirmPassword)) &&
        validator.equals(password, confirmPassword)) {

        console.log("User email of: " + email + " is valid.\n");
        console.log("User first_name of: " + firstname + " and last_name of: " + lastname + " are valid.\n");
        console.log("User password of: " + password + " and confirmPassword of: " + confirmPassword + " are valid and are matching.\n");

        next();
    }
    else {

        if(!(validator.isEmail(email))) {

            res.status(400).send({
                error: "Enter a valid email"
              });
        }

        if(validator.isEmpty(firstname) ||
           validator.isEmpty(lastname) ||
           validator.isEmpty(username) ||
           validator.isEmpty(password) ||
           validator.isEmpty(confirmPassword)) {

            res.status(400).send({
              error: "One or many of the inputs you wrote are empty, please try again"
            });
        }

        if(!(validator.equals(password, confirmPassword))) {
            res.status(400).send({
                error: "Your password and confirm password need to match"
              });
        }
    }
},
(req, res) => {
    store
        .createUser(req.body.firstname, req.body.lastname, req.body.email, req.body.username, req.body.password)
        .then((createdUser) => {
            console.log(createdUser);

            store.createUserCart(createdUser.user_id)
            .then((userCart) => {
                console.log(userCart);
            })
            .catch(error => {
                console.log(error);
                res.status(500).send({ error: "Unabe to create user shopping cart" });
            });

            res.status(201).send(createdUser);
        })
        .catch(error => {
            console.log(error);
            res.status(500).send({ error: "Unable to insert the user" });
        });
});
```

Then, within .createUser() in store.js, we encrypt the password which was passed in as a parameter, write our SQL statement involving `INSERT` syntax, and then apply the data we'd like to add into the database with the parameters we've provided .createUser(). The encryption process is done on line 88 using the bcrypt library. All we have to do is specify the number of salt rounds for our encryption. `saltRounds` is just a variable which is set equal to 10. Then, we pass our password parameter and the number of salt rounds into bcrypt's .hash() function, which will create our encrypted password. At this point, we have validated the user registration data which is why we move towards inserting the data into our users table. In regards to error handling, if an error occurs with the insertion of a new user, we are notified of this if the length of the result[0] object is less than one. If the result[0] object is less than one, then this means there was not a new row created in the users table.

```
86  async function createUser(firstname, lastname, email, username, password) {
87
88    const encPassword = await bcrypt.hash(password, saltRounds);
89
90    const date = new Date();
91
92    const result = await pool.query(
93      "INSERT INTO users SET first_name = ?, last_name = ?, email = ?, username = ?, password = ?, created_at = ?, is_active = ?, is_buyer = ?, is_seller = ?",
94      [firstname, lastname, email, username, encPassword, date, 1, 1, 0]
95    );
96    if (result[0].length < 1) {
97      throw new Error(
98        `Failed to create a new user ${username}`
99      );
100   }
101   return getUserById(result[0].insertId);
102 }
```

5. Self-Check: Adherence to original non-functional specs

Non-functional Requirements Specifications	Status	Comment
1. There should be WebSocket functionality for displaying the most recently created posts, the hottest daily pics, and other displays of item data. a. WebSocket functionality should be used for displaying user's notifications, messages, and for auctions.	Issues	Low priority in relation to implementing our superior feature price matching
2. SSL certificate, which enables websites to move from HTTP to HTTPS, for better website security.	On track	
3. Warning/caution list page for what sellers/buyers should look out for. A standard list providing details and warnings for potential scammers or bots.	On track	
4. Submit a ticket. After flagging an item since it seems sketchy, or an item might be fake, the user could continue the process by filing a ticket and sending a message to us to follow through further with reporting suspicious behavior.	On track	
5. Limited login attempts for security purposes. For example, if the user enters the wrong password 3 times, they will be locked out of trying to log in to their account for 30 minutes.	Issues	Low priority over implementing input validation for user login
6. Captcha to fight against bots. Captcha provides an extra level of security for users.	On track	
7. Email verification for both seller and buyer. Verification ensures that there is a valid user behind each seller and buyer account.	Issues	Low priority over implementing functional requirements
8. Two factor authentication for added security measures. The user will be required to authenticate their account on two different devices to help prevent a breach in security.	Issues	Low priority over implementing functional requirements
9. Use of sessions on the backend to securely keep track of the user's shopping cart information once they redirect from	Done	

the main shopping site.		
10. Users should be able to view and update their personal information settings securely. a. Users can update their account information, change their full name, update their email, change their password. Update their payment information or see their complete payment history in a secure fashion.	Done	
11. Password recovery for all users when they need it. The recovery process will involve having an email sent to the user to the email address associated with their account, where they would have the opportunity to change their password.	Issues	Low priority over implementing functional requirements
12. Username recovery for all users when they need it. Similar to the password recovery process, where the user will get an email to recover their username.	Issues	Low priority over implementing functional requirements
13. Load balancers or distributed microservices on the backend so that we don't have just only servers doing all of the backend work? We don't want the backend to go down and have the whole server unavailable to all users.	Issues	Low priority over implementing functional requirements
14. Keeping track of unaccept/accepted usernames and write reviews. a. They must be appropriate and cannot contain duplicate usernames.	Issues	Low priority over implementing functional requirements
15. If the user does not do anything while logging in for like 15 minutes or more, then it automatically logs out the user.	Done	Done but user session logs out after 30 minutes
16. Correct login information entered. Username and password matching correctly.	Done	
17. Valid post information when a seller creates a new post.	Done	

18. The user's shopping cart should be updated with the total amount of the user’s cart and their selected items FROM THE SERVER. User shopping cart data cannot be handled by the client, the client should only be used to display the data sent from the server.	Done	

19. Username should be at least 8 characters in length. The username shall have a minimum length for security reasons.	On track	
20. User password should contain uppercase/lowercase and numbers with a minimum length of 8 characters. This password requirement ensures an extra layer of security.	Issues	Low priority over implementing functional requirements
21. When a user inputs their email into the registration form, a valid email should have @ along with a valid ending. This requirement checks if the email is in the right format.	Done	
22. New post titles should not exceed 80 characters in length.	On track	
23. New post description should not exceed 500 characters in length.	On track	
24. A minimum of 10 salt should be used when hashing a user's password.	Done	
25. Perform password hashing when a user creates a new account. This will guarantee us that passwords saved in the database are secure.	Done	
26. Bcrypt library should be used for hashing a user's newly created password.	Done	
27. A user's session should be serialized after logging in, and deserialized when a user leaves the site.	Done	
28. Stripe.js or paypal payment SDK should be used for creating and validating payments from users.	Done	
29. Seller analytics page (demographics of customers, what is sold more frequently, etc.) Analytics will provide the seller with information to help them better their product listings.	Issues	UI implemented but functionality is low priority
30. Passport library should be used to authenticate an existing user's requests.	Issues	Low priority in relation to functional requirements
31. Encapsulate any SQL statements so that SQL injections are not possible.	Done	
32. The table for seller posts should have	Done	

the following constraints for its ID column: integer type, auto increment, and public key.		
33. Each post should have a foreign key tied back to the user's id who created the post.	Done	
34. Each post should have columns for the posts' title, description, price, photo,the time when the post was created, if the post is still active on the app or whether it has expired.	Done	
35. Post comments should be represented in a comments table. Each comment's id should be of integer type, auto incremented, and the public key.	On track	The table for product comments is created but product comments are not finished
36. Each comment should also have a foreign key tied back to the creator of the comment, and a foreign key tied back to the post the comment was for. Each comment should also have the time that it was created.	Done	
37. Session data for each user should be stored in the database. If their session is still valid, allow them to log in directly into the app.	On track	
38. Redux should be used to manage the state for the react client. Redux will be essential for keeping track of user's shopping cart information.	Done	
39. Use of redux mapStateToProps() for updating user's shopping cart information, losing user account details. a. The mapStateToProps() function will be used to load state data, such as a user's shopping cart items as properties passed into React components so that the data can be displayed to the user.	Done	
40. Multer library should be used for uploading images to the server/database.	Done	
41. Browser support for IE, Chrome, Firefox, Brave, etc. Cross-browser support for all users to be able to access.	Done	
42. By default, users who are not logged in should still be able to access the	Done	

marketplace. It will show up on the top right corner that they are not logged in/login is shown instead of logout.		
43. When fetching data from api routes, there should be a maximum response time of 1s. Anything more than this will impede upon user experience.	Done	
44. When a user sends an axios request, the server should be able to parse through urlencoded or Json data.	Done	
45. React components should be kept simple. Design a React component so that it accomplishes one job for the user. If a component is more complex, then use multiple react components with parent/child relationships.	Done	
46. An object titled INITIAL_STATE should be used in each Redux reducer to hold each reducer's state data.	Done	
47. Redux actions should start with the name of the reducer in capitals, an underscore, plus whatever the action is doing, an underscore, plus whatever the action is changing. Example: USER_UPDATE_POST, USER_SEND_MESSAGE, etc.	Done	
48. If a team member has to create a new React route, then do so in App.js. This is where all of the Routing takes place.	Done	
49. Variable names should not be redundant, but instead describe what role the variable has. Thinking of good naming conventions is important so everyone else on the team knows why and how the variable was used a. By default use camel case for variable names.	Done	
50. Commenting is essential. Before jumping into code, every team member should write some form of pseudocode describing what it is they're going to code before actually doing so. a. Outside of pseudocode, commenting should be used everywhere to describe how	Done	

features or architecture works.		

51. At least 3 images of products required on listing. Having more than 1 image of the product provides customers with more information of how the product will look.	Done	
52. Minimum/maximum size of images on product page. An appropriate size of the images (perhaps 600x600 or 2x2) will help users with product visibility.	Done	
53. At least a 10 word description of the product being sold. The seller will provide adequate details on the product listing.	Done	
54. Buyer protections provide buyers with peace of mind when using the app.(Paypal has 180 days for users to be refunded if there is an issue with the purchased item).	Issue	Low priority compared to implementing functional requirements
55. At least a 5 word title of the product being sold. The seller will provide an adequate title on the product listing.	Done	
56. Once an item is sold, the listing is automatically updated to reflect as such. This will notify users that the listing is no longer available/has been sold, or that the number of products available is now different.	Issue	Low priority compared to implementing functional requirements
57. Products must be concrete, not abstract (no services, only physical items.) Restrictions for what can be bought and sold on the app should be made clear.	Done	
58. Maximum number of listings per seller- 100? 1000? To keep balance of how we are trying to cater to the smaller businesses.	Done	
59. Offer commitment on each listing that has the offer option. Once the buyer makes an offer or bid and wins, a transaction automatically occurs and the buyer gets their account charged (automatic withdrawal.)	Issue	Low priority compared to implementing functional requirements
60. Product page must include how much of each item is available. The number of products available lets the buyer know how much is in stock.	Done	
61. Product page has an expiration of 60 days. The expiration ensures that the	Issue	Low priority compared to implementing

		functional

product listings are current and that sellers are active.		requirements
62. Sellers should not be able to reject buyer's meetup time within 2 days, unless it is an emergency case.	Issue	Low priority compared to implementing functional requirements
63. Product page must include a place of origin. This requirement will let the buyer know where the product is coming from, and perhaps what to expect when it comes to shipping time.	On track	
64. Product page must include if the item is new or used. This detail is essential to let the buyer know the condition of the product being sold.	On track	
65. 24 hour window to modify order; after the 24 hours, buyers will be committed to their order.	Issue	Low priority compared to implementing functional requirements
66. Must show the seller's name of a selling item.	Done	
67. Must show the seller's rating selling items.	Issues	Low priority compared to implementing functional requirements
68. Stay connected- follow us on social media. This will provide the app with cross platform exposure and potential business.	Issues	Low priority compared to implementing functional requirements
69. App news/announcements. Users will have the option to subscribe to the app to receive newsletters through email.	Issues	Low priority compared to implementing functional requirements
70. Paid ads throughout the site. Advertisements will be another source of revenue for the app. Perhaps the end of the page will include a link titled "advertise with us" to attract potential business.	Issues	Low priority compared to implementing functional requirements
71. FAQ page that highlights all the questions and answers that users generally have when using the app.	On track	
72. Coupon/promo code expiration. Every code will have an expiration date to keep discounts as special.	Issues	Low priority over implementing functional requirements

73. Recently sold items on the home page. This will show users what has recently sold, and for how much it has sold for.	Issues	Low priority compared to implementing functional requirements

74. Team members should use pm2 as their process manager when running the project locally. ‘npm start’ is NOT necessary for running the project. Only ‘pm2 start process.config.js’ should be used.	Done	
75. Freenom website should be used to configure a free domain name for our app.	Done	
76. If the AWS EC2 instance is stopped and then reset, the public ipv4 address will have changed. This means the app’s public IP address must be changed in /credentials, and it must be updated in freenom’s domain name configuration.	Done	
77. Team members should use the ‘systemctl status nginx’ command to test the status of the nginx web server.	Done	
78. If necessary, team members should only update the nginx server blocks contained in /etc/nginx/sites-available/default. No where else.	Done	
79. In order to verify any errors related to nginx, always use the command ‘sudo nginx -t’	Done	
80. If team members need to reload the nginx server, then run ‘sudo systemctl reload nginx’.	Done	
81. If team members pull changes to a remote branch, make sure to ‘npm install’ so you have any new packages/libraries installed from other people’s commits.	Done	
82. Team members should always create new branches from the development branch.	Done	
83. After you are done with your work, create a merge request from your branch back into the development branch so we can compare the differences between the two.	Done	
84. If you ever have a question about any new code pushed to github, then team members must ask about what they’re unsure on in the discord #help channel.	Done	

85. When team members encounter a bug in the code, team members should write down the steps in order to duplicate the bug, then create a pull request labeled as 'bug' so that the rest of the time can replicate the bug.	Done	
86. If team members need to add additional documentation for a feature they've created, they should add the documentation label onto their pull request.	Done	
87. If a team member is creating a new feature, then they should mark their pull request with the enhancement label.	Done	
88. If a team member has a question about a portion of code they are writing, then they should create a pull request from their branch back into development with the question label. This way, the rest of the team can review what the question is to resolve it.	Done	
89. Code should be merged from the development branch into master branch only when all merge requests for the current build have been agreed upon and added to the current git history.	Done	
90. Whenever a milestone is close to submission, a thorough check of the /credentials folder should be done to ensure everyone has the most recent updates.	Done	
91. SSH should always be used when logging into the remote EC2 instance.	Done	
92. Whenever logging into the EC2 instance, make sure to check for updates: use the following commands: 'sudo apt update', 'sudo apt upgrade' or 'sudo apt full-upgrade', and 'sudo apt autoremove'. If you are prompted to reboot, then reboot the instance.	Done	
93. Create a CI/CD pipeline for deploying code to the remote EC2 instance.	Issues	Low priority over implementing functional requirements

94. The Node api server should have access to react build files so it works correctly on the remote EC2 instance.	Done	

95. In order to login to EC2 instance correctly, team members should ensure that the csc648.cer file in their local github repository's / credential folder has file permission 600.	Done	
96. CSS styling to create a box-shadow around an item when the user hovers over it. This will allow the user to view the item clearly.	Done	
97. High contrast between test and background for ease of use. This will make the overall appearance of the app easy to view.	Done	
98. The sidenav should be implemented with the React transition group library.	Done	Finished but implemented without the react library since this was not needed
99. All Contact, About, Careers, company information, etc. should be stored in the footer of the app.	Done	
100. All Login/Logout, Shopping Cart, Notification, Auction pages should be stored in the sidenav menu.	Done	
101. @media queries should be used for making the app usable on mobile devices.	On track	
102. In order to satisfy requirements for screen sizes <= iPhone 5, the minimum width for @media should be 300px.	Done	
103. In order to satisfy requirements for screen sizes <= iPhone 6, the minimum width for @media should be 360px.	Done	
104. In order to satisfy requirements for screen sizes <= iPad, the minimum width for @media should be 720px.	Done	
105. In order to satisfy requirements for screen sizes <= desktop, the minimum width for @media should be1280px.	Done	
106. By default, buyers will not have their shipment information added. If they choose delivery, then they must be prompted for their address information.	Done	
107. Users should be able to choose what page route that he or she needs to go to such as Home, Notification, About,	Done	

Contact, Login, and Logout.		
108. A hamburger menu should be used to represent the navigation bar, this menu should have an animation which opens and closes it.	Done	
109. Users should be able to interact with a “See More” at the end of the top 50 posts, which when clicked will load another 50 posts to view on the Home page.	Issues	Low priority compared to implementing functional requirements
110. When users click the hamburger menu, there should be a navigation bar component which is triggered in addition to the side nav which should pop out from the side.	Done	
111. Navigation bar should have an animation triggered when it is opened or closed. a. This task should be implemented with some sort of CSS transform or the use of a React component to represent the navigation bar.	Done	
112. When a user clicks the Notifications tab in the navigation bar, a dropdown menu should appear. a. A list of the user’s notifications ordered by most recent to least recent is a requirement.	Issues	Low priority compared to implementing functional requirements
113. When a user clicks on a notification from another user, they should be redirected to a page displaying their conversation history.	Issues	Low priority compared to implementing functional requirements
114. If a user is not logged in, views an item, and then clicks the button to message the seller, the user should be prompted to either login with their username/password or to create an account with a button.	Issues	Low priority compared to implementing functional requirements
115. Lock password input in login from being updated or clicked on after the user clicks again.	Issues	Low priority compared to implementing functional requirements
116. If a user attempts to login and they use the wrong credentials, they should be redirected back to /login route.	Done	
117. Once a user logs in with the correct	Done	

username/password, redirect them to the home page at route '/' to start interacting with our app's features.		
118. Users can switch to dark or white background mode for the app.	Issues	Low priority compared to implementing functional requirements
119. Category menu (beauty/cosmetics, home, appliances, tech gadgets/computers, athletic tools, clothing). This will allow searching items faster, and to keep all the products organized.	On track	
120. Photo slider which has left/right arrows allowing the buyer to navigate through the pictures of an item posted by a seller. This will allow the buyer to see details on the design of the product.	Issues	We had time for implementing one image, multiple images became a priority 2 which is why the photo slider has not been finished.
121. Have an option to slide through the posts or go to the next page.	Issues	Low priority compared to implementing functional requirements
122. Display seller's contact information.	Done	
123. Shopping cart with number of items icon. The buyer will be able to view the total items in their cart and the total cost of the items.	On track	
124. Option to drag item(s) to the shopping cart button. This will provide shopping ease for buyers.	Issues	Low priority compared to implementing functional requirements

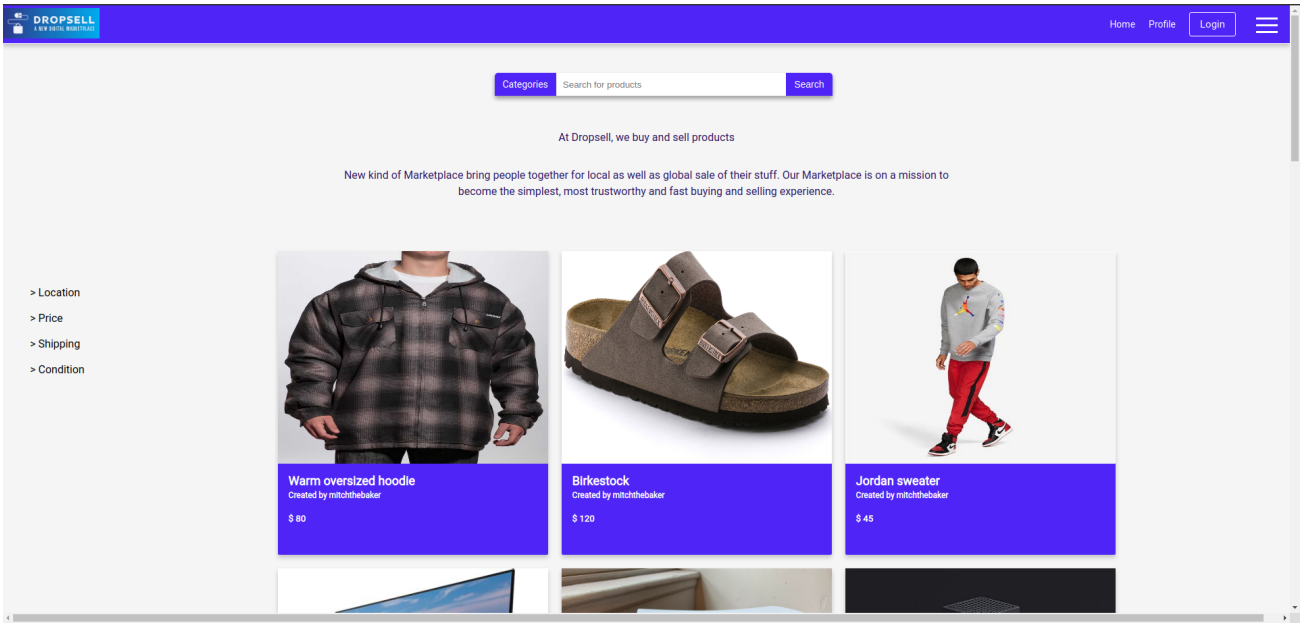
6. List of Contributions

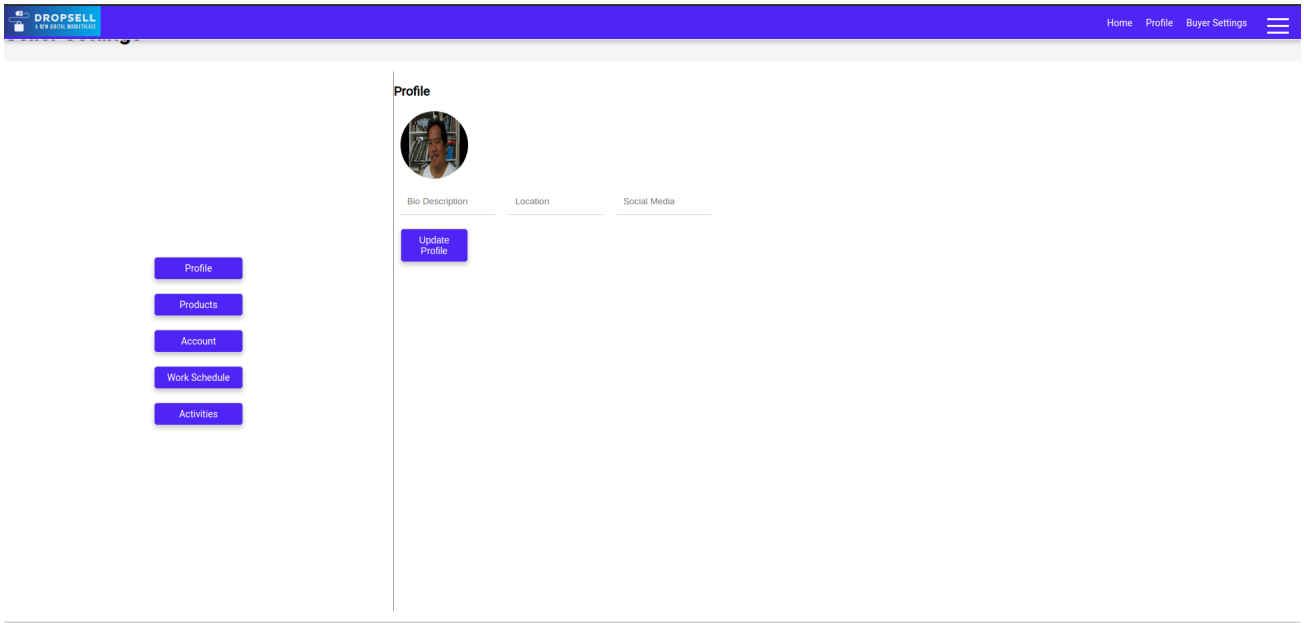
Student Name	Contributions
Mitchel Baker	Mitchel was responsible for writing the usability test for the creation of new marketplace products. He was also the main contributor for the self-check on best practices for security section, while also working with Rowena to finalize the adherence to nonfunctional requirements. In addition to the Milestone 4 document, he also continued implementation of our team's functional requirements for the application.
Charmaine Eusebio	Charmaine was one of the main contributors for the QA test plan section. She collaborated extensively with Krina to perform valuable QA testing for our product's main feature.
Kenneth N Chuson	Kenneth was responsible for writing up the code review section of the Milestone 4 document. After he finished writing the code review, he had Mitchel review the work he had submitted for constructive feedback. He also wrote the usability test plan for buyer/seller scheduling.
Krina Bharatbhai Patel	Krina was one of the main contributors for the QA test plan section. She collaborated with Charmaine to write up a test plan which played an important role in breaking apart the main section of our product's main feature. Her work helped our backend team in coming up with an efficient plan for implementing our product.

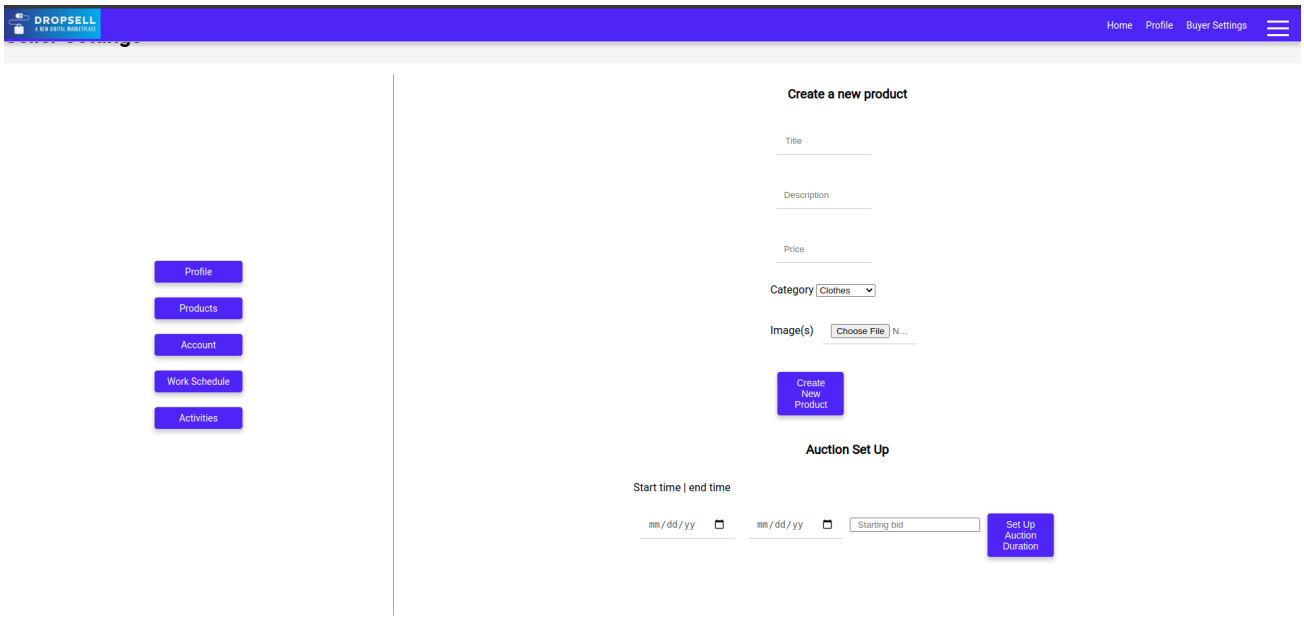
Michael Schroeder	Michael played a key role in the following usability test plans: creation of an auction product, adding products to cart, and for price matching. He created the usability test and Lickert Scale questionnaire templates for our team which was

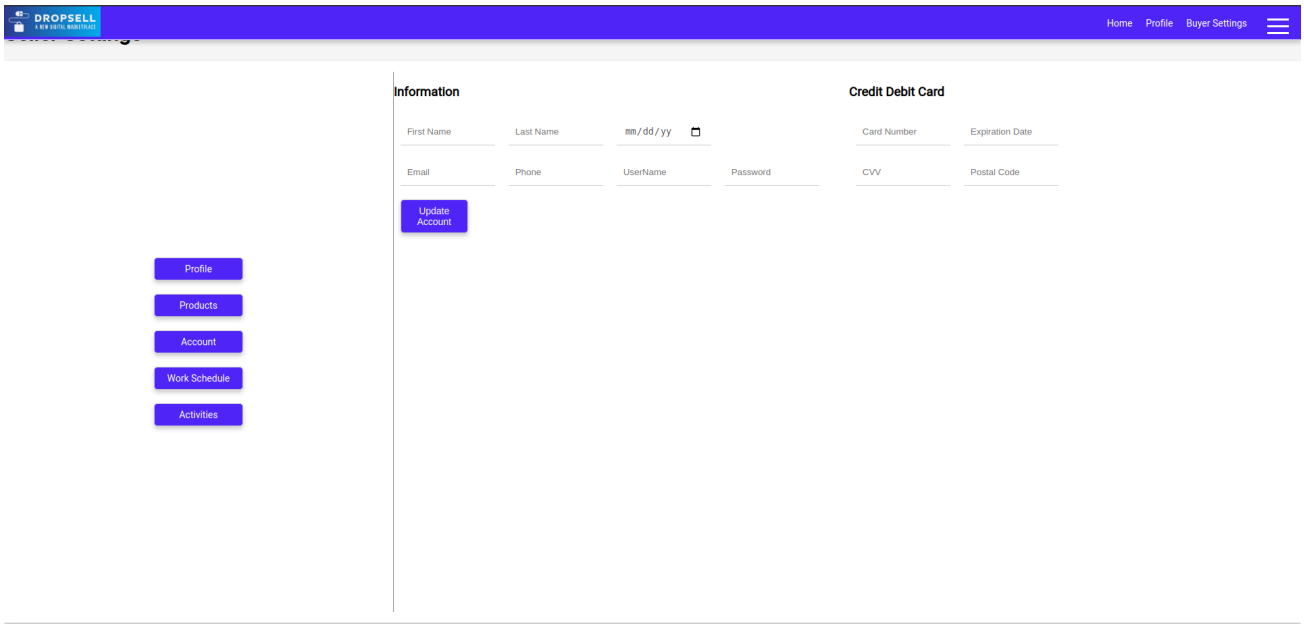
	extremely helpful for everyone else writing usability test plans.
Rowena Elaine Echevarria	Rowena played a key role by creating the table for the self-check on adhering to our original nonfunctional specifications. She communicated effectively on the requirements she was not sure about, and collaborated with Mitchel by working together on finalizing which requirements were done, on track, or had issues.
Jamie Dominic Walker	Jamie was the lead contributor to our product summary section in the Milestone 4 document. He applied precision and care by taking note of our main priority one requirements, while rewording each function in a way that was coherent and readable. He communicated to the rest of the team by requesting feedback when finished, which demonstrates his consideration for the rest of his team members.

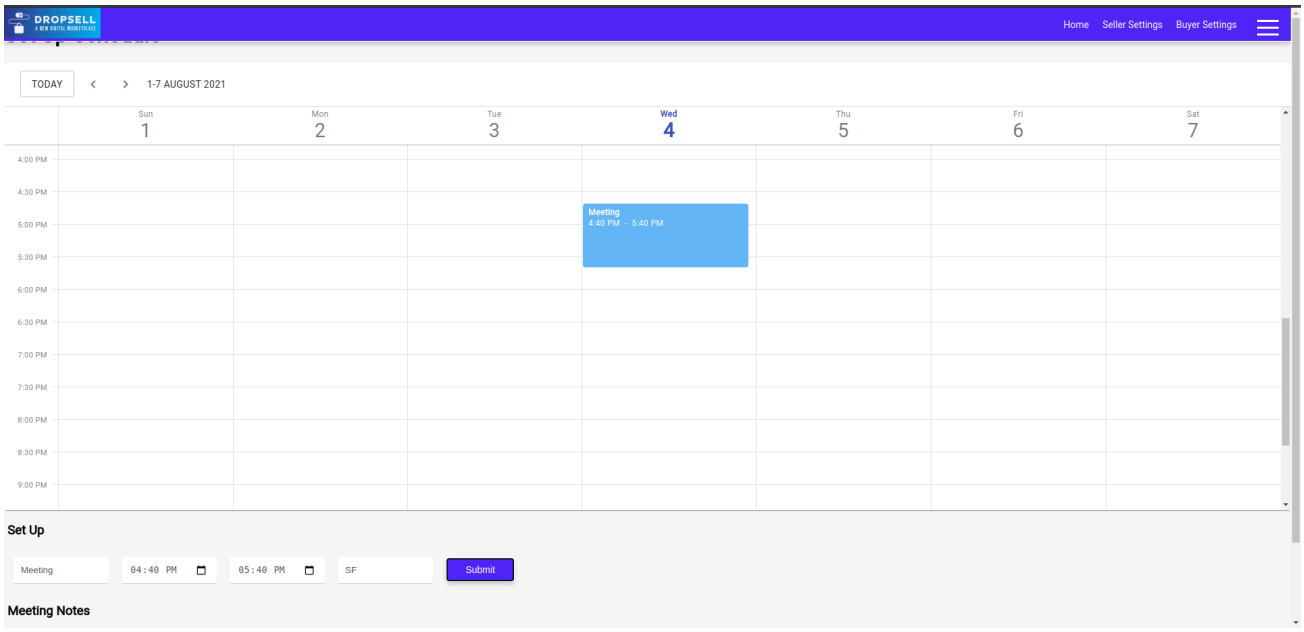
4. Screenshots of actual final product as shown in demo



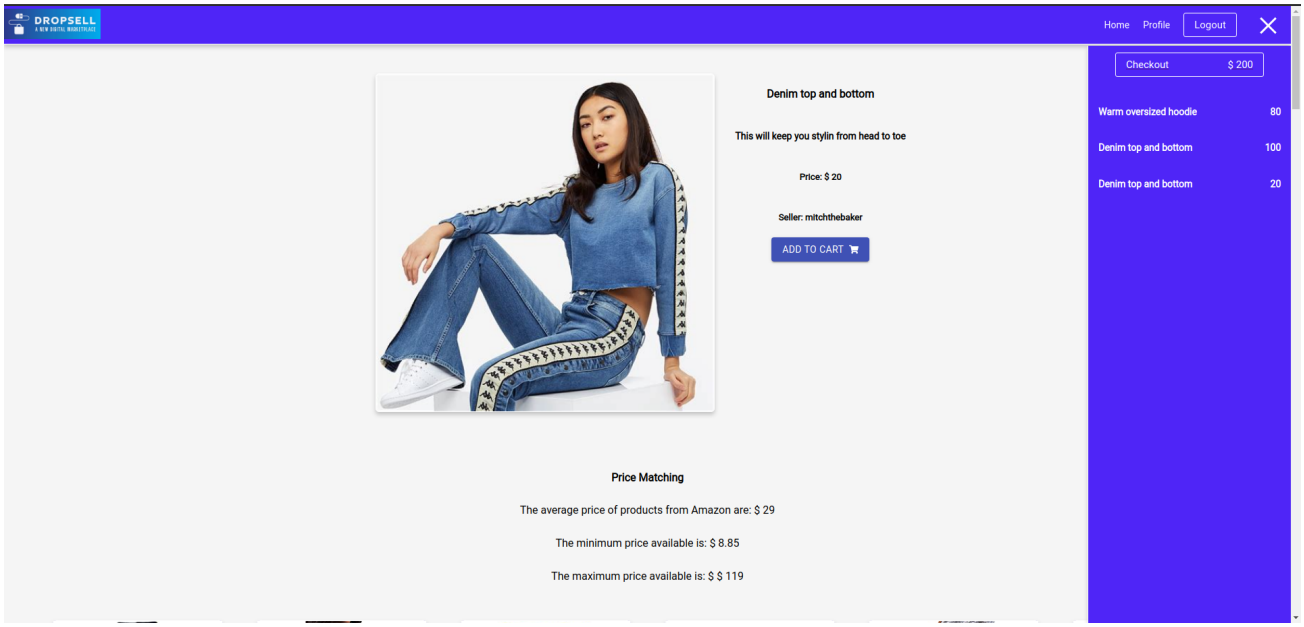


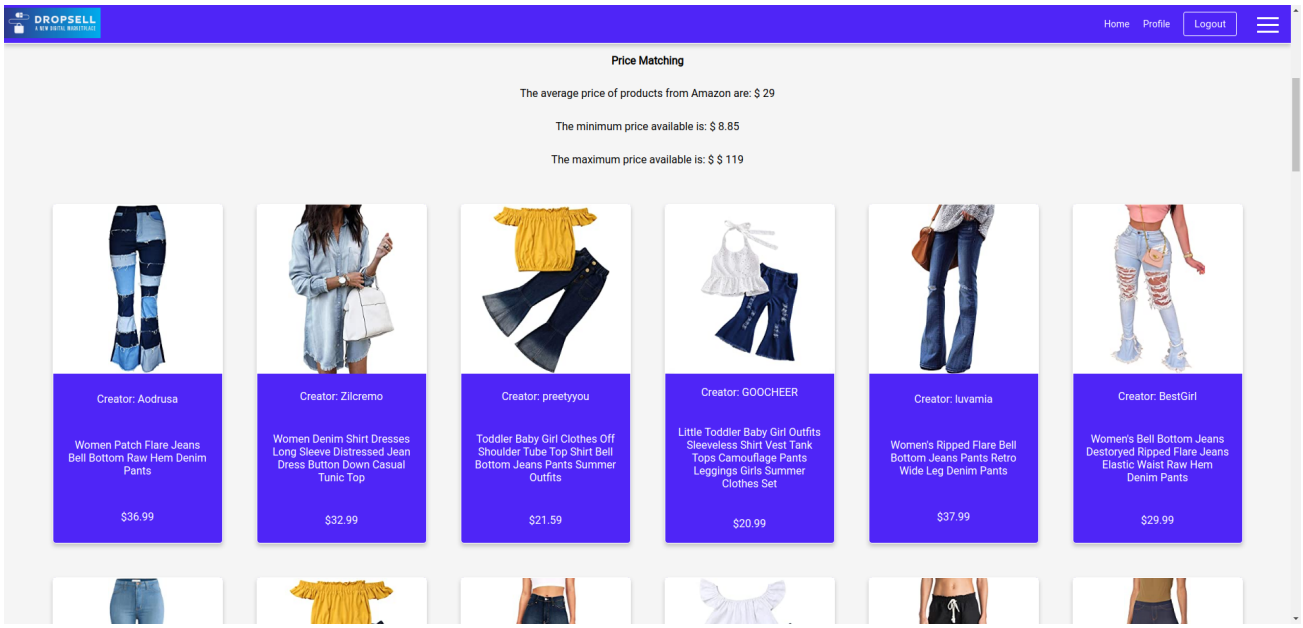

DROPSELL
Home
Profile
Buyer Settings
Profile
Products
Account
Work Schedule
Activities
Create a new product
Title
Description
Price
Category
Clothes
Image(s)
Choose File
Create New Product
Auction Set Up
Start time | end time
mm/dd/yy
mm/dd/yy
Starting bid
Set Up Auction Duration

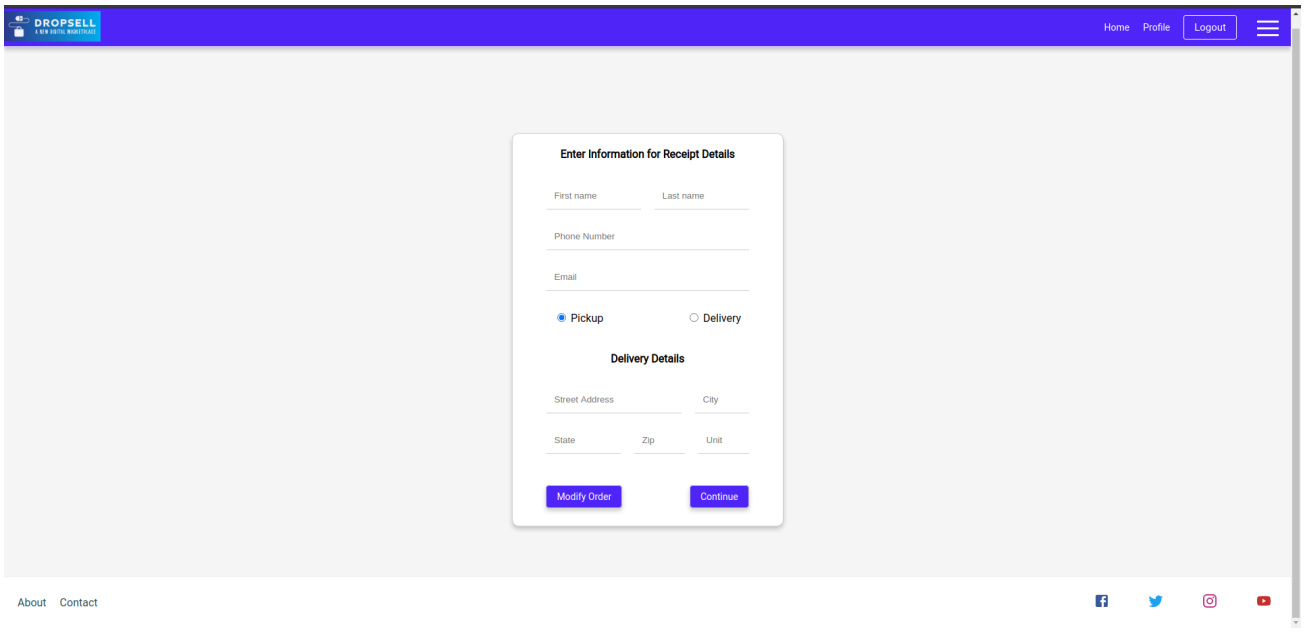

DROPSELL
Home
Profile
Buyer Settings
Profile
Products
Account
Work Schedule
Activities
Information
First Name
Last Name
mm/dd/yy
Email
Phone
UserName
Password
Update Account
Credit Debit Card
Card Number
Expiration Date
CVV
Postal Code

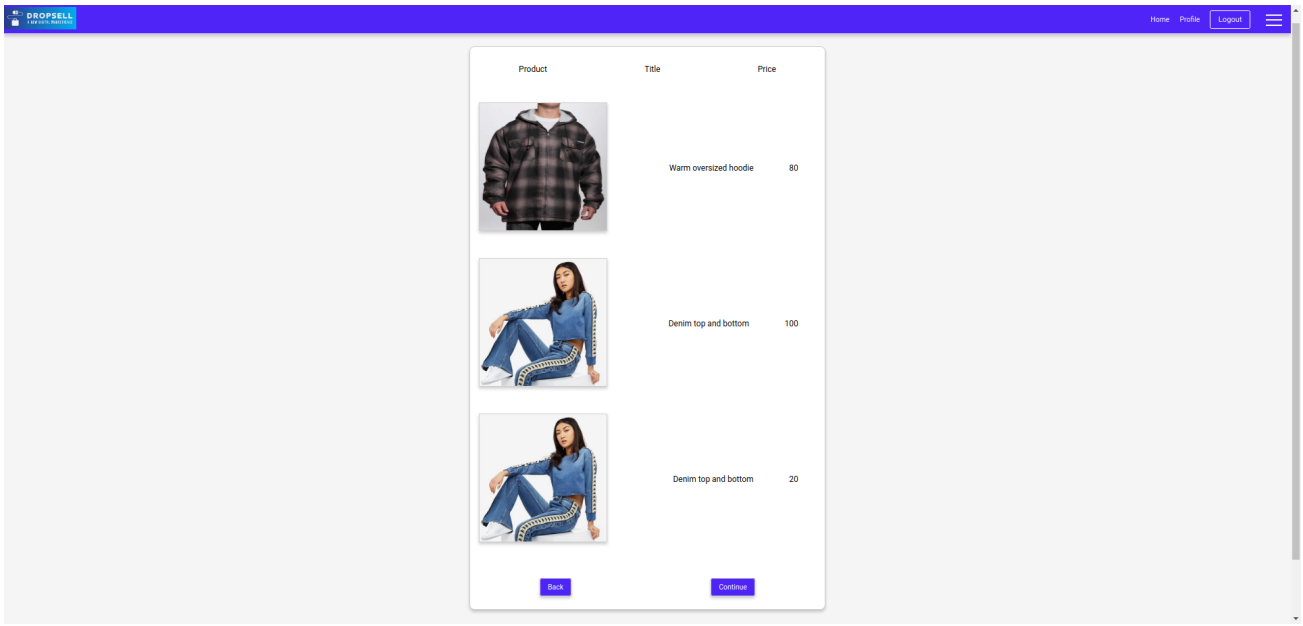

DROPSELL
Home
Seller Settings
Buyer Settings
TODAY
1-7 AUGUST 2021
Sun 1
Mon 2
Tue 3
Wed 4
Thu 5
Fri 6
Sat 7
Meeting
4:40 PM - 5:40 PM
Set Up
Meeting
04:40 PM
05:40 PM
SF
Submit
Meeting Notes

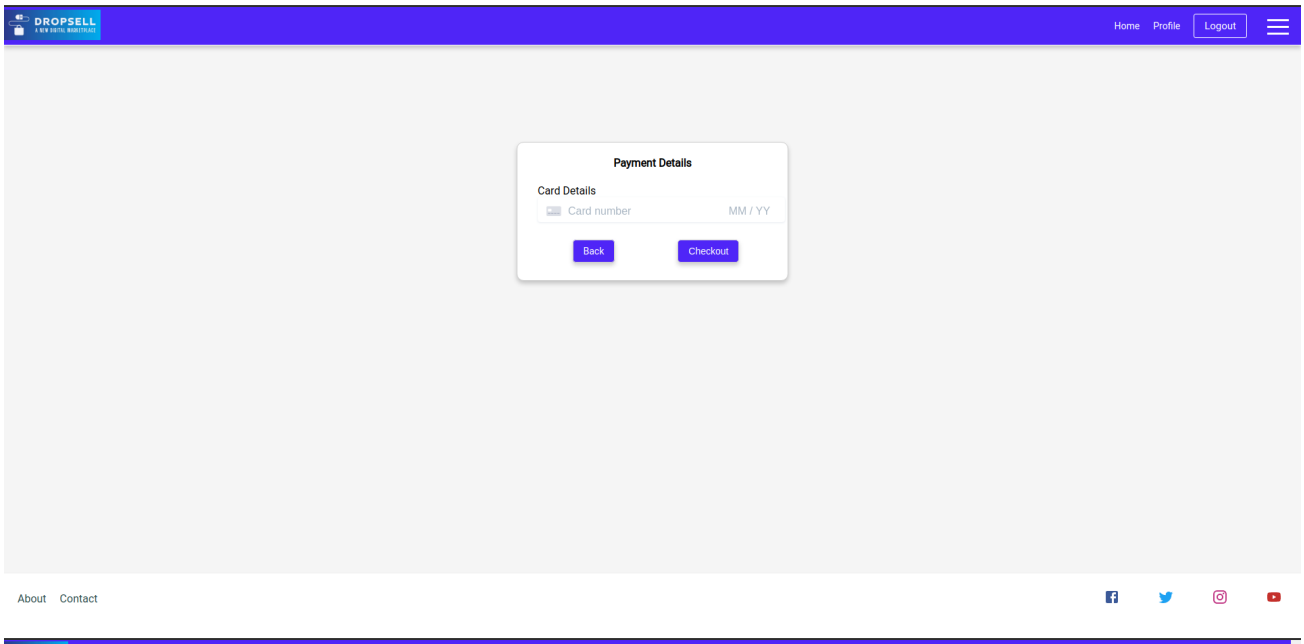

DROPSELL
Home
Profile
Buyer Settings
Profile
Products
Account
Work Schedule
Activities
# of Activities
undefined
Sells
Posts
Refunds
Returns
Reports

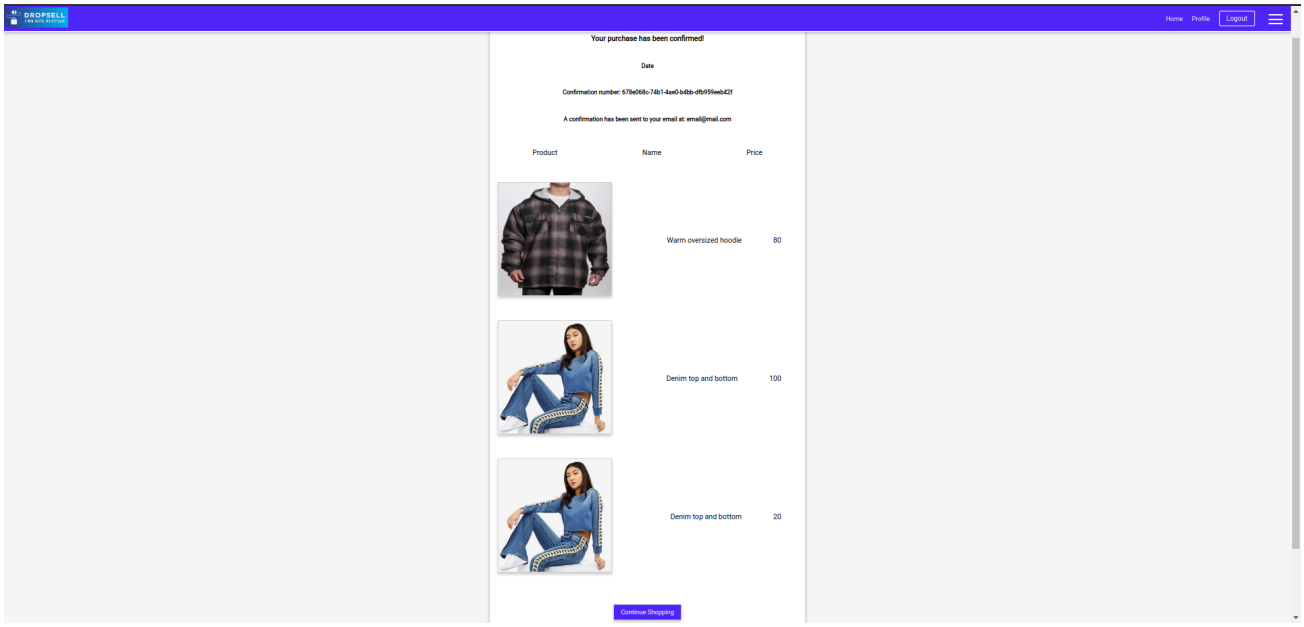

DROPSELL
Home
Profile
Logout
Checkout
$ 200
Warm oversized hoodie
80
Denim top and bottom
100
Denim top and bottom
20
Denim top and bottom
This will keep you stylin from head to toe
Price: $ 20
Seller: mitchthebaker
ADD TO CART
Price Matching
The average price of products from Amazon are: $ 29
The minimum price available is: $ 8.85
The maximum price available is: $ $ 119


DROPSELL
Home
Profile
Logout
Price Matching
The average price of products from Amazon are: $ 29
The minimum price available is: $ 8.85
The maximum price available is: $ $ 119
Creator: Aodrusa
Women Patch Flare Jeans Bell Bottom Raw Hem Denim Pants
$36.99
Creator: Zilcremo
Women Denim Shirt Dresses Long Sleeve Distressed Jean Dress Button Down Casual Tunic Top
$32.99
Creator: preetyyou
Toddler Baby Girl Clothes Off Shoulder Tube Top Shirt Bell Bottom Jeans Pants Summer Outfits
$21.59
Creator: GOOCHEER
Little Toddler Baby Girl Outfits Sleeveless Shirt Vest Tank Tops Camouflage Pants Leggings Girls Summer Clothes Set
$20.99
Creator: luvamia
Women's Ripped Flare Bell Bottom Jeans Pants Retro Wide Leg Denim Pants
$37.99
Creator: BestGirl
Women's Bell Bottom Jeans Destoryed Ripped Flare Jeans Elastic Waist Raw Hem Denim Pants
$29.99

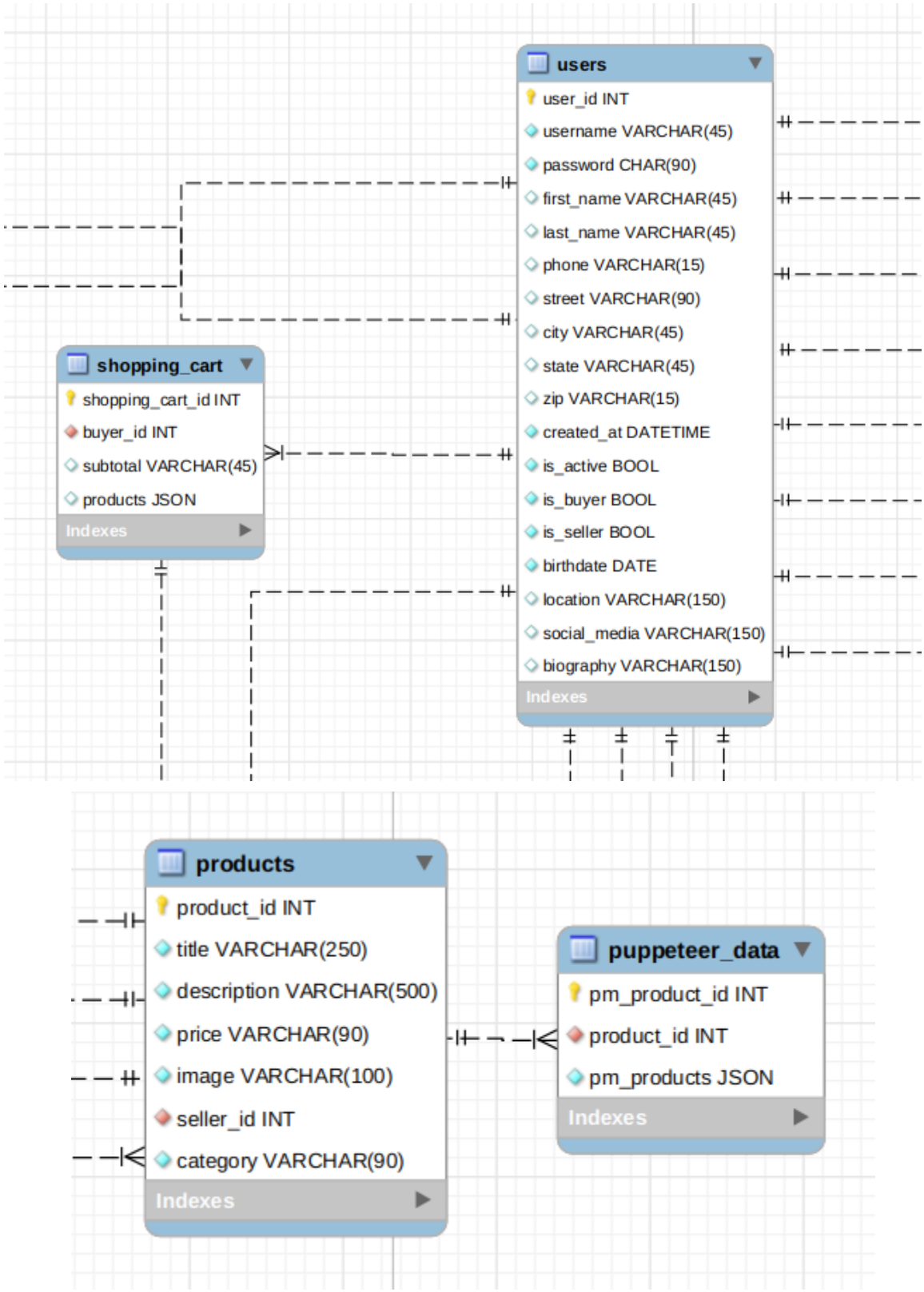

DROPSELL
Home
Profile
Logout
Enter Information for Receipt Details
First name
Last name
Phone Number
Email
Pickup
Delivery
Delivery Details
Street Address
City
State
Zip
Unit
Modify Order
Continue
About
Contact


DROPSELL
Home
Profile
Logout
Product
Title
Price
Warm oversized hoodie
80
Denim top and bottom
100
Denim top and bottom
20
Back
Continue


DROPSELL
Home
Profile
Logout
Payment Details
Card Details
Card number
MM / YY
Back
Checkout
About
Contact

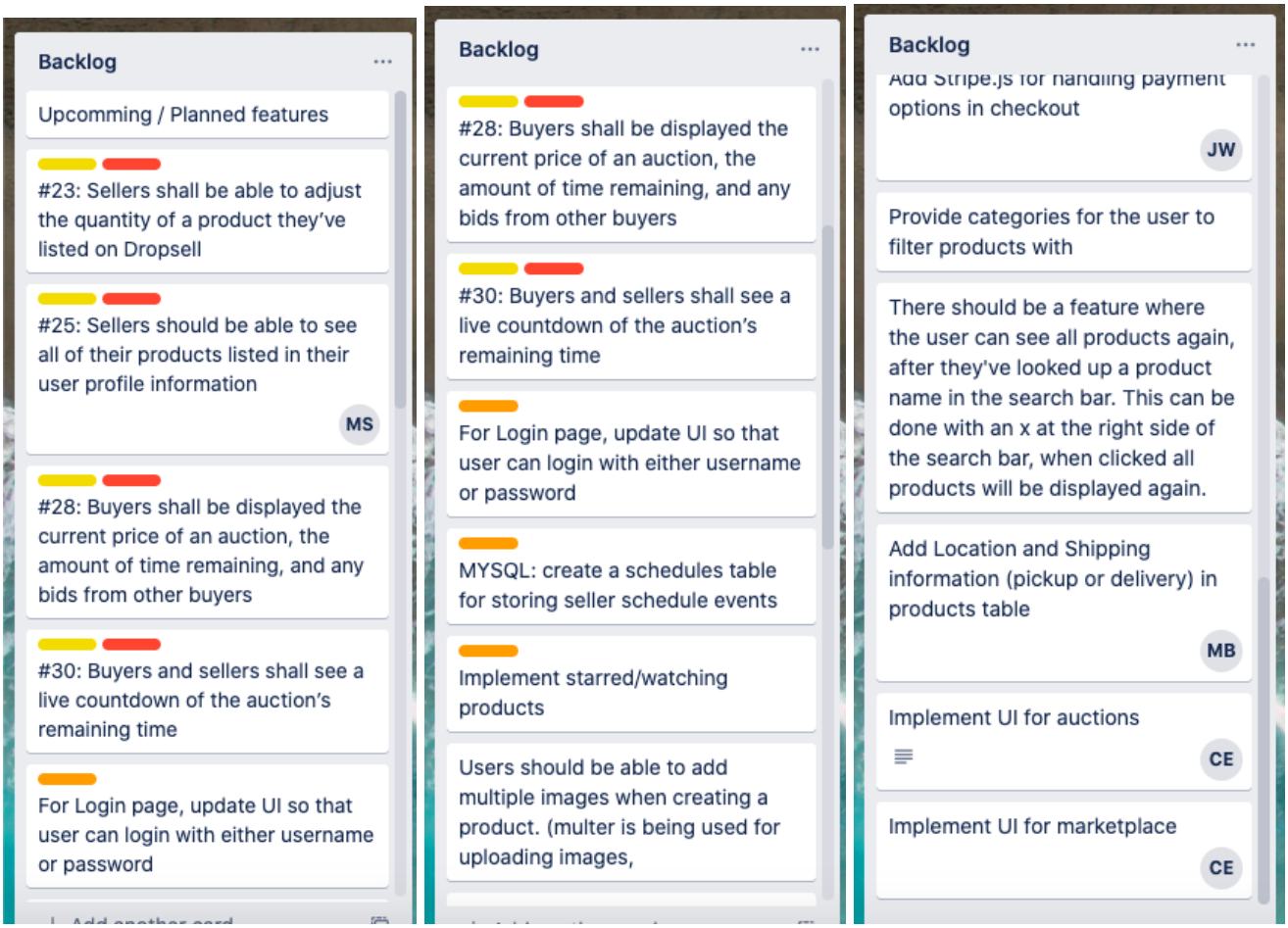

DROPSELL
Home
Profile
Logout
Your purchase has been confirmed!
Date
Confirmation number: 678e068c-74b1-4ae0-b4bb-dfb959eeb42f
A confirmation has been sent to your email at: email@mail.com
Product
Name
Price
Warm oversized hoodie
80
Denim top and bottom
100
Denim top and bottom
20
Continue Shopping

5. Screenshots of main DB tables

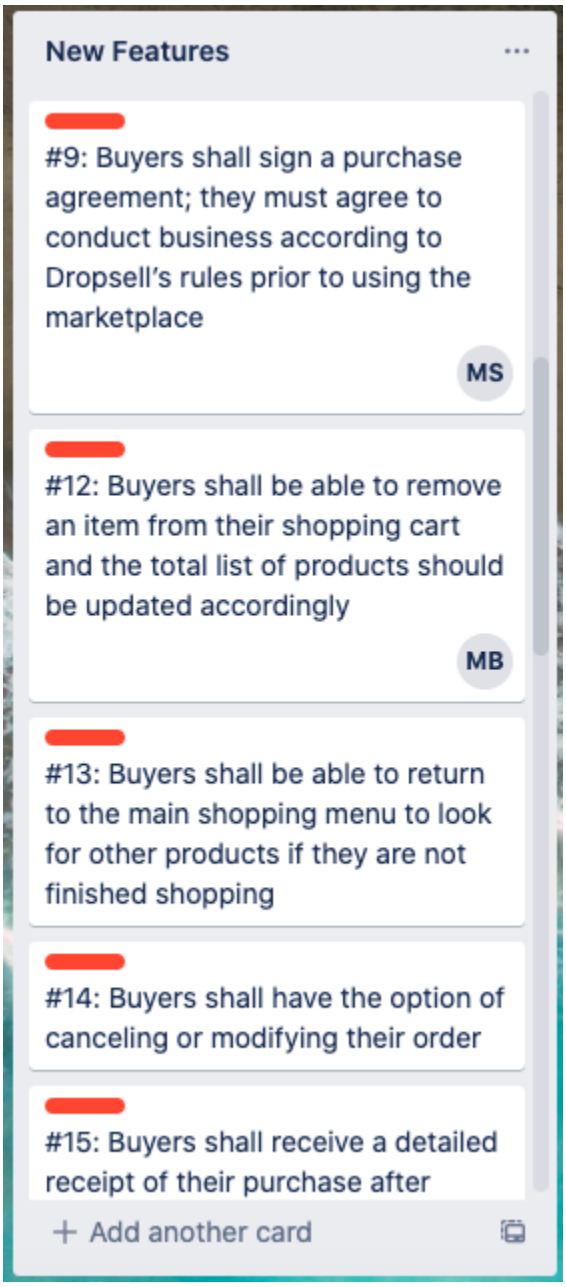


6. Screenshots of Trello

Backlog

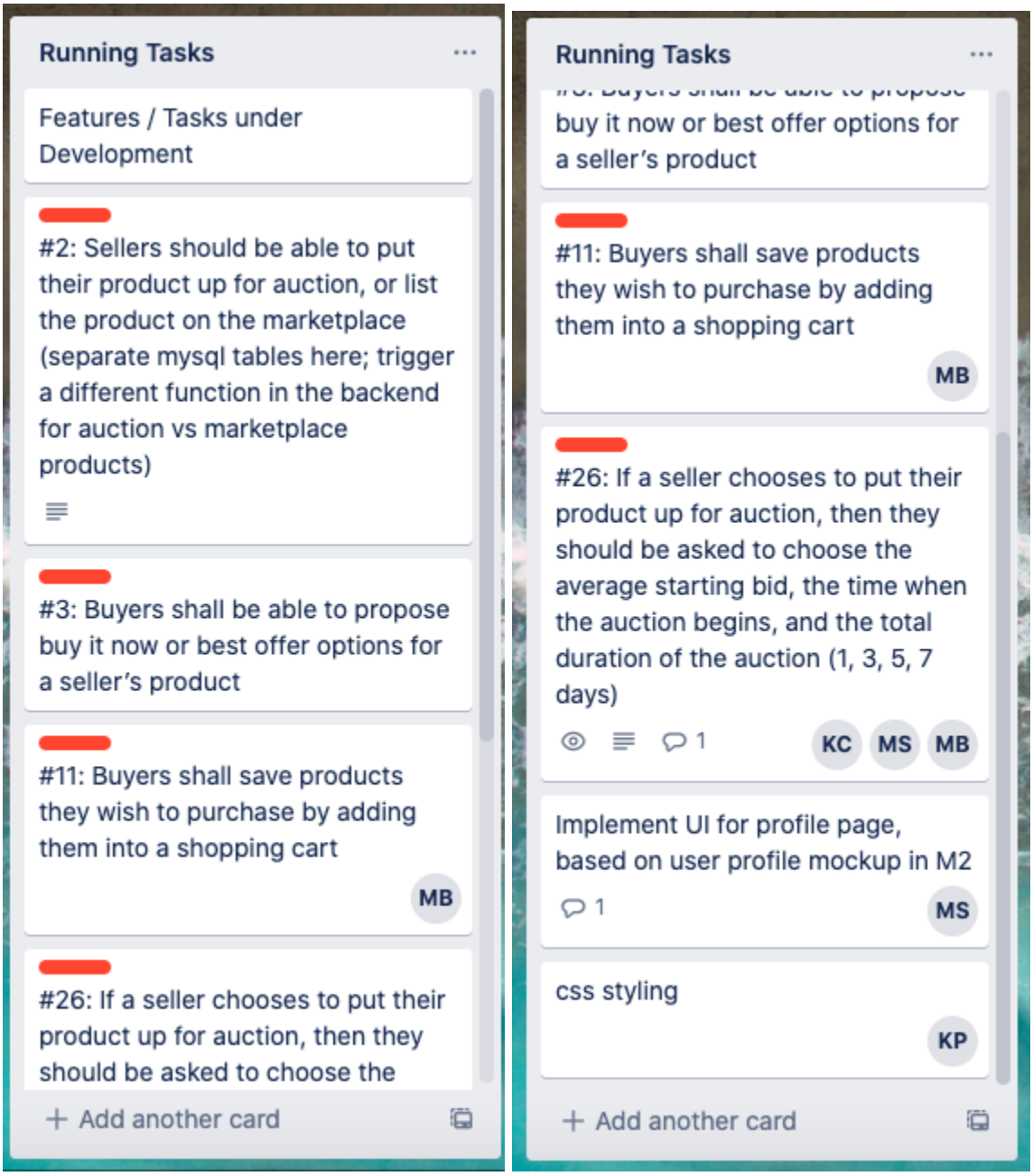


New Features

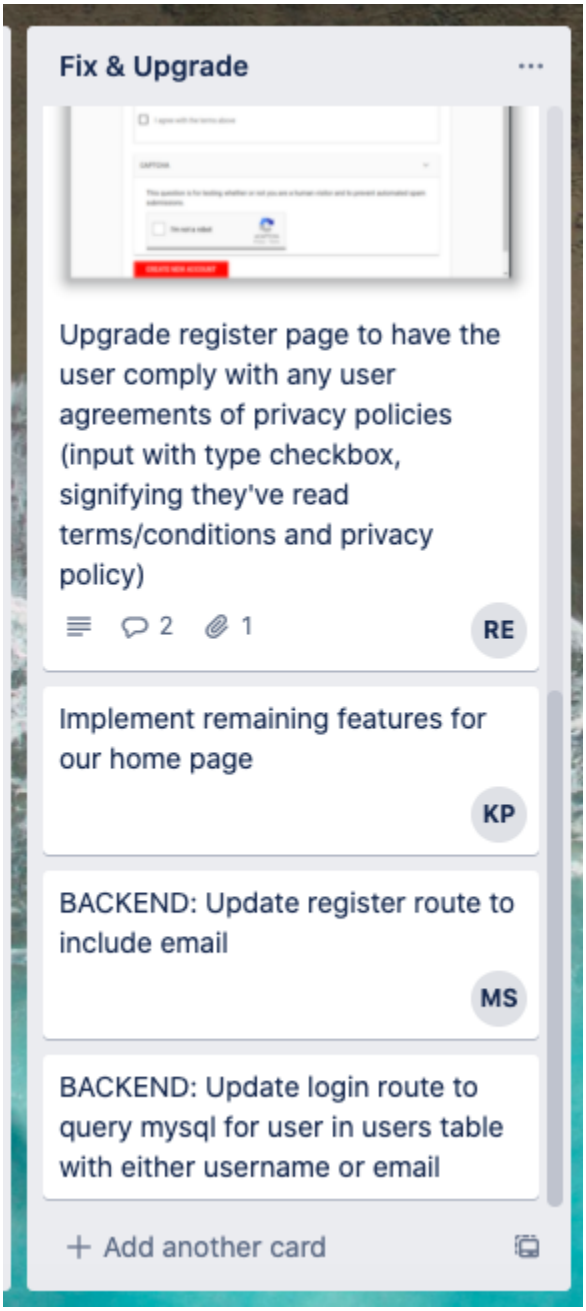


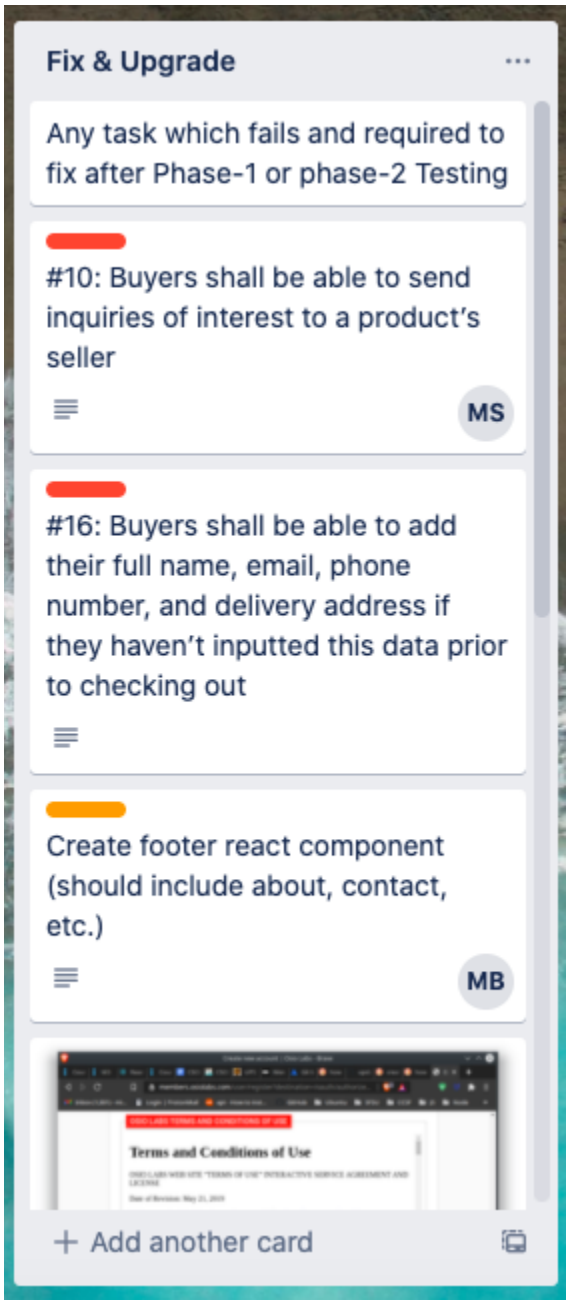

New Features
#15: Buyers shall receive a detailed receipt of their purchase after checking out
#17: Buyers shall have the option of choosing an existing payment option or adding a new one
#19: Sellers shall sign a contract before being granted the privilege of posting products on Dropsell
MS
#22: Sellers should be able to edit the title, description, price, and image of their products (PUT method with updated values in req.body or in req.params)
KC
#24: Sellers should be able to delete products from Dropsell (GET
Add another card
New Features
#24: Sellers should be able to delete products from Dropsell (GET method with user id to delete in req.params, then call SQL statement to delete the product row in the products table)
#29: Buyers shall be able to bid on products with one click
#31: Buyers shall have the ability to contact sellers directly through the Dropsell website
Add sessions when user logs in, first by implementing sessions with express-sessions, then use mysql-express-sessions library as a wrapper to save sessions to db
MS
MB
Create the auction page using React components and Redux state management
Add another card

Running Tasks

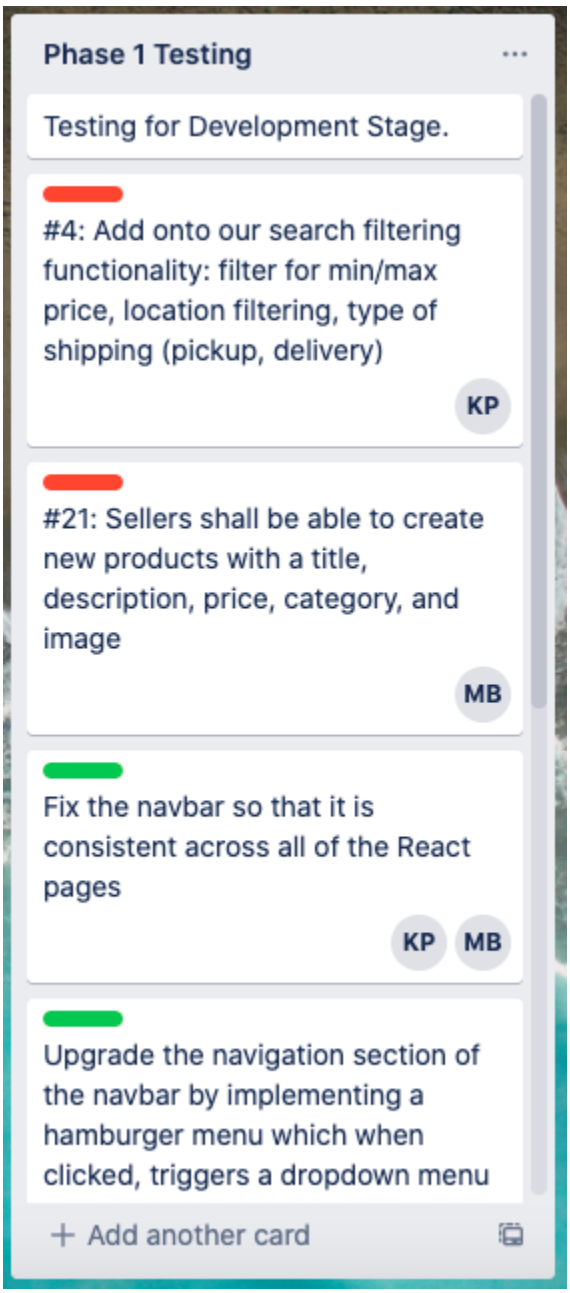


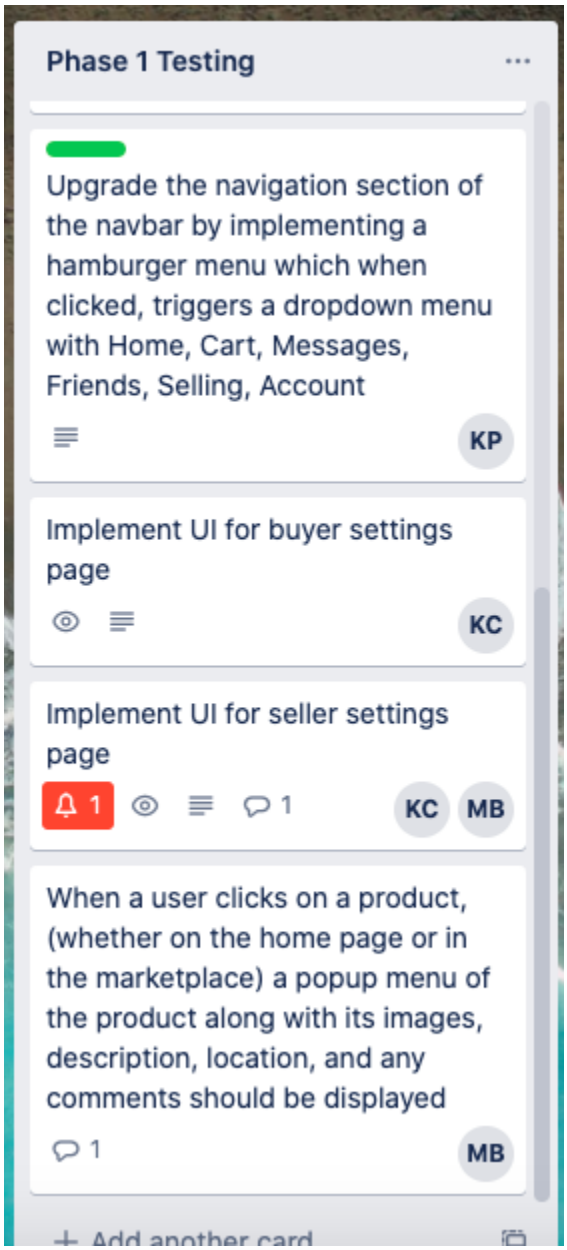
Fix & Upgrade


Fix & Upgrade
Upgrade register page to have the user comply with any user agreements of privacy policies (input with type checkbox, signifying they've read terms/conditions and privacy policy)
2
1
RE
Implement remaining features for our home page
KP
BACKEND: Update register route to include email
MS
BACKEND: Update login route to query mysql for user in users table with either username or email
+ Add another card

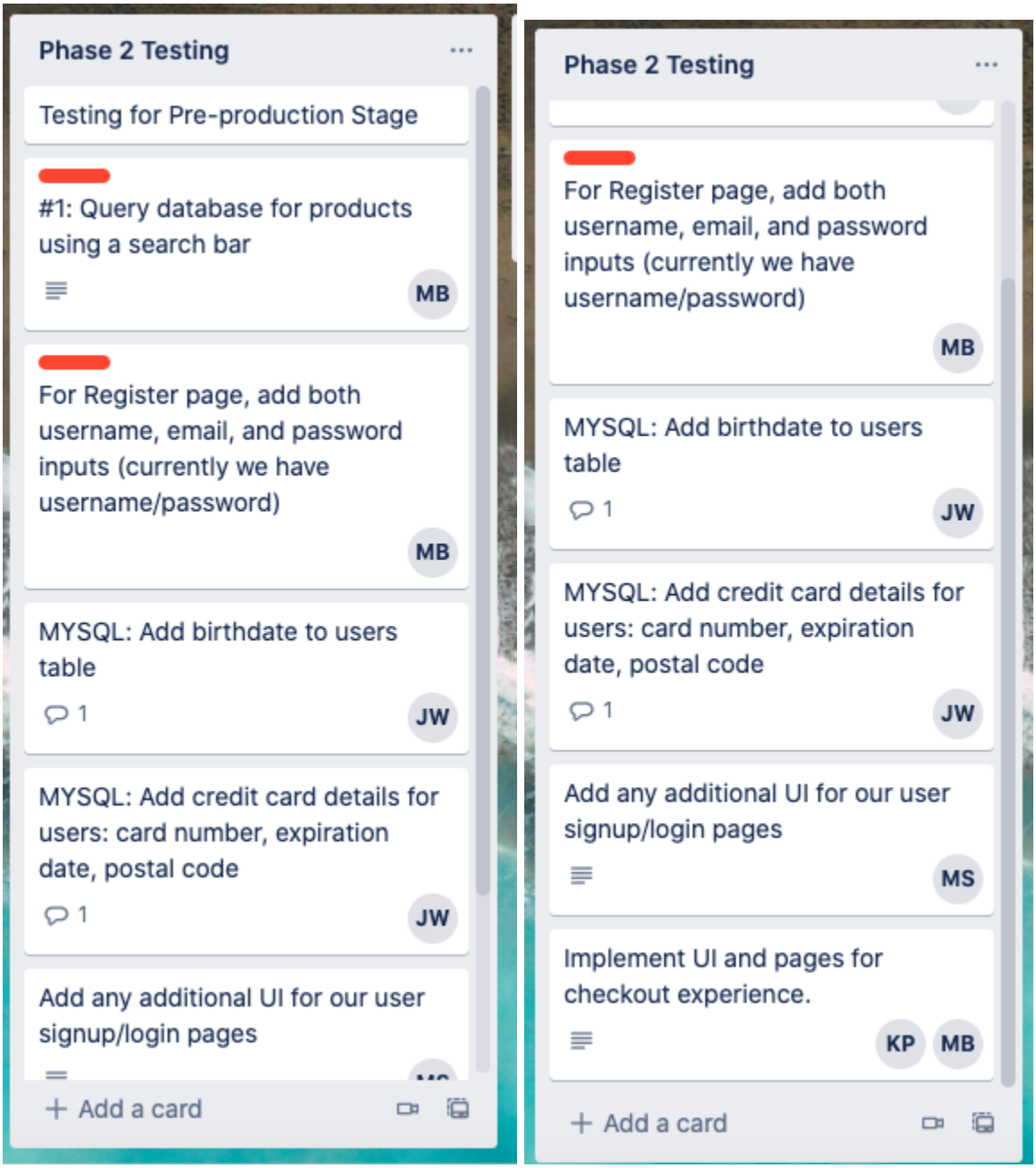

Fix & Upgrade
Any task which fails and required to fix after Phase-1 or phase-2 Testing
#10: Buyers shall be able to send inquiries of interest to a product's seller
MS
#16: Buyers shall be able to add their full name, email, phone number, and delivery address if they haven't inputted this data prior to checking out
Create footer react component (should include about, contact, etc.)
MB
Terms and Conditions of Use
+ Add another card

Phase 1 Testing

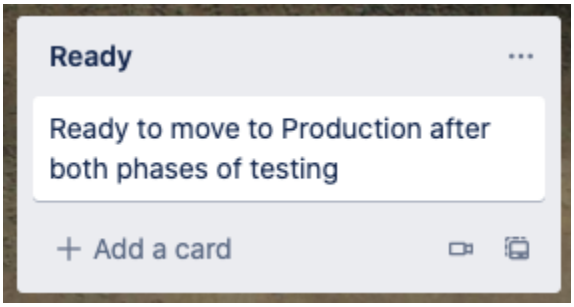




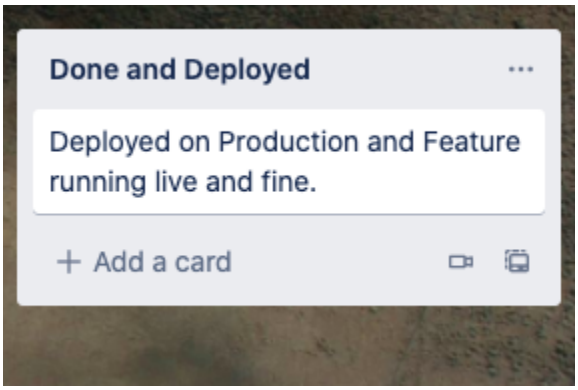
Phase 2 Testing



Ready



Done and Deployed



7. Team contributions

Team Contributions	
Team member name	Score
Krina Bharatbhai Patel	8
Charmaine Eusebio	6
Rowena Elaine Echevarria	6
Michael Schroeder	8
Kenneth N Chuson	10
Jamie Dominic Walker	6

8. Post analysis

One of the first challenges our team faced was scheduling times where our whole team could be present. We were still able to find time, at least a couple times per week, where everyone could be present at a single meeting. But oftentimes, we would have team-specific meetings with the backend team or frontend team to discuss their respective areas of the project. Of course, team-specific meetings were always open to all team members, but they were optional for people who weren't directly contributing to tasks related to the meeting agenda at hand. Scheduling was only an issue towards the beginning of the course, ultimately, we all adjusted to each other's availability, and we found a simple fix for being able to have synchronous meetings together. Outside of scheduling, I would say the main limiting factors for our team were balancing exterior responsibilities and obligations such as work, social events, family time, plus handling other personal circumstances while also having to juggle multiple summer courses in addition to CSC648. Many of us are working part-time/full-time roles in addition to taking summer courses, and all of us have family and friend obligations to attend to despite having to work/study consistently every day for the whole Summer. A couple of us were also in the process of moving into different homes, which soaks up time due to planning, having to check out new places, coordinating with new housemates, etc.

All of us knew what we were getting into when signing up for a summer course, especially CSC648. We all knew it would require a ton of dedication and hard work in order to get to the end, so there is no excuse about completing work on time even if we have to balance work, social life, family time, personal circumstances. I think if I could change anything for myself to address this challenge, I would have ideally liked to have taken CSC648 during the Spring or Fall semester. Due to scheduling reasons and wanting to graduate by the end of this Fall, that wasn't an option, but I think taking Software Engineering for a full semester is the most ideal option. I know my team members were also in similar situations where they're trying to knock out multiple classes during the summer semester to either graduate early or get ahead with course work. But honestly, signing up for three or four summer classes in addition to trying to live life is close to physically impossible since there are only 24 hours each day to work with. It is really important for us, in the future, to think ahead and plan accordingly the workload we'd like to place upon ourselves and to also think about other life circumstances which will take up our time. I think this would also improve our quality of learning if we focused on a couple classes during a summer session as opposed to taking three or more.

Regarding learning new topics this semester, it is important to emphasize that software engineering is an extremely fast paced environment. Oftentimes, we will not have exposure to a specific technology which requires us to learn it on the spot, self-teach ourselves the best practices for implementing the said technology, and executing accordingly. It may seem unrealistic to expect this of software engineers, but this is the reality. I think that our project was a good example of having to get up to speed with a full stack application very quickly. As we all know, we created the about pages and set up our deployment stack the first week, then had our vertical/horizontal prototypes up and running within a month. Another challenge I think our team faced was troubleshooting the steep learning curve of having to incorporate two separate applications, the frontend React application and the backend Node application while also incorporating the third piece, a mysql database.

In addition to all this, the fourth piece is setting up the deployment server and creating our devops infrastructure. There was a ton to pick up, which was why I felt like it was my responsibility as a team lead to abstract most of the complex pieces of code so that our team could focus on implementing code in the frontend React and backend Node applications. Of course, I can't blame any of my teammates for not learning fast enough. This is not a rational idea, since everyone's had two months to get up to speed with our tech stack. I would like to emphasize though that in a real project, it will be expected of you to get up to speed independently. You will have to make mistakes, look things up when you hit a wall, troubleshoot bugs on your own, and problem solve all while combining critical thinking skills into a full stack web application with multiple working pieces. Our solution to the learning curve issue was doing code demos during our meetings.

By doing a hands-on demo of how our application communicated internally, I think this really helped our team to better understand how everything was connected. We were also open to any and all questions, and quick to respond to questions, so our team could get immediate feedback with the issues they were facing. When I was working on projects in prior internships, I did not always have a reliable support line for getting questions answered. As a result of that, I wanted to answer all questions immediately, so everyone had the experience of making mistakes but also learning from these mistakes and improving their programming knowledge and expertise.