Team 16 - Product Backlog

Rohith Ravindranath, Kenny Varghese, Raman Kahlon, Spencer Kline, Ashwin Mahesh

Table of Contents

Purpose

With the development of the Internet, we have constantly been finding new ways to make our lives easier. The advent of e-commerce sites has made it so easy to buy a product/service that, instead of going to a store, you can order a product without even leaving the comfort of your home.

Many e-commerce websites have come to the forefront of the Internet. However, not all of them are created equal. We intend to create a site which provides an excellent client experience regardless of whether the user is a merchant or a customer. We also plan on incorporating a Business to Business sale scheme, something that other e-commerce companies do not offer. Other prominent aspects we will perfect are recommended products, reviews, and search autocomplete. With these properties, we aim to build a web application that can compete with top sites like Amazon and eBay.

Design Outline

High Level Overview

For our project design we will use two main frameworks, Firebase for backend and Angular for frontend development. Firebase will provide us with a solid foundation, with real-time databases to store and access user information. Angular works well with Firebase and provides a strong structure for a wide range of aspects, including designing the UI. For the front end, we will be coding in JavaScript, a high-level programming language used in most websites, along with HTML/CSS.

1. Client

   In our project, the client will be the interface that users can utilize to interact with our e-commerce website. The client functions by sending Ajax requests to our server and receiving the responses from the server. The client will then reload the webpage with the appropriate changes to the user interface.

2. Server

   As a result, our server will receive requests from the user. The server then takes the request and, if necessary, will send queries to the database to either retrieve, modify, or add data on request. The server then compiles the accurate request response to send back to the appropriate user.

3. Database

   The database will store all the data needed for the website. This includes the user information, product information, and merchant information. The database receives queries from the server and, in turn, sends query responses of the extracted data back to the server.

Design Issues

Functional Issues:

1. What information do we want when users register as a customer?
   a. **Email, Password, Name**
   b. Email, Password, Name, Phone Number
   c. Email, Password, Name, Phone Number, Address

   Choice: option A

   Justification: When registering an account, the password and name are vital to identification of users. Also, adding an email will allow for extra verification of identity and allows for email confirmations on purchase. The users phone number and address are not required for verification because email will suffice. In conclusion, the email, password and name will be enough for registering.

2. How do we calculate the average review of products?
    a. Flat average
    b. Add weight to trusted users

    Choice: option B

    Justification: Ratings on a product play an important role in the decision process of making a purchase by the user. For this reason, it is vital that our rating system is very representative of the product. That is why we have chosen to average the review rating of products by adding weight to trusted users and then averaging with that factor. This allows for a more accurate representation of the products review rating for the use of the user.

3. How do we calculate what products to recommend?
    a. Find products that have the most matched tags
    b. Show highest-rated products with most matched tags

    Choice: option B

    Justification: Recommendations of products play an integral aspect in supporting the pursuit of the products users really want. Consequently, we need to find the perfect products to show our users. Therefore, we have chosen to calculate what products to recommend by showing the highest-rated products that have the most matched tags. This creates a more accurate system of matching users with the perfect products.

4. How should we assign tags to products?
    a. Ask questions and compute tags
    b. Allow merchant to assign tags

    Choice: option B

    Justification: Again, recommendations of products are a very integral aspect of getting users the best matched products. Therefore, it is important that the tags of products are accurately representative of the product itself. Consequently, we have chosen to allow merchants to assign their own tags on products

5. How do we verify merchants?
    a. **Annual fee, like Apple Developer Program**
    b. Allow anyone to be merchant
    c. One-time fee
    d. Thorough manual inspection of potential merchant

    Choice: option A

    Justification: The verification of merchants is an important aspect of our marketplace because we want to hold a high standard of products for our users to create a better shopping experience. Therefore, we want to make sure that our merchants are annually reviewed and meet our standards. For scalability purposes it is too difficult to expect a thorough manual inspection of potential merchants to be efficient enough. Therefore, we have chosen to have an annual fee for merchants.

6. How should products be initially sorted?
    a. Alphabetically
    b. Date posted
    c. Rating
    d. Most viewed product within time frame

    Choice: option C

    Justification: By sorting products by rating, we are designing our website to cater to the users. This way, the high-quality products that users want are prioritized so that they can easily view and purchase products that others are buying and approving of.

Non-Functional Issues:

1. What web service?
    a. AWS
    b. Heroku
    c. Firebase

    Choice: option C

Justification: Since we decided on using Firebase as our backend framework, we felt it would be best to use their built-in deployment as that would be the most compatible and easiest to make live.

2. What language do we want to use?
    a. **JavaScript**
    b. Python
    c. Java
    d. Swift

Choice: option A

Justification: The language that we have chosen to move forward with is JavaScript. With JavaScript, we can choose a stack and use JS as the language for both front-end and back-end. It is also an easy to learn and powerful language that many of us in this group already know.

3. What backend framework should we use?
    a. **Firebase**
    b. Django
    c. Node
    d. Spring

Choice: option A

Justification: We wanted real-time features in our application which put us between Node and Firebase. Through our own research, we found Firebase to be an easy to pick up and powerful framework that we wanted to explore. Thus, we are moving forward with Firebase.

4. What database should we use?
    a. MySQL
    b. MongoDB
    c. **Firebase**
    d. Redis

Choice: option C

Justification: When a user (customer/merchant) registers on our website, we need a way to store and access their information efficiently. Firebase utilizes real-time databases to form a system that can accurately keep track of user login information, as well as payment options for customers, such as saving a credit card number to an

account to allow fast and effective transactions. This way, data can be synced across all clients in real time and is still available if the site goes offline.

5. What front-end framework should we use?
    a. **Angular**
    b. React
    c. Jinja

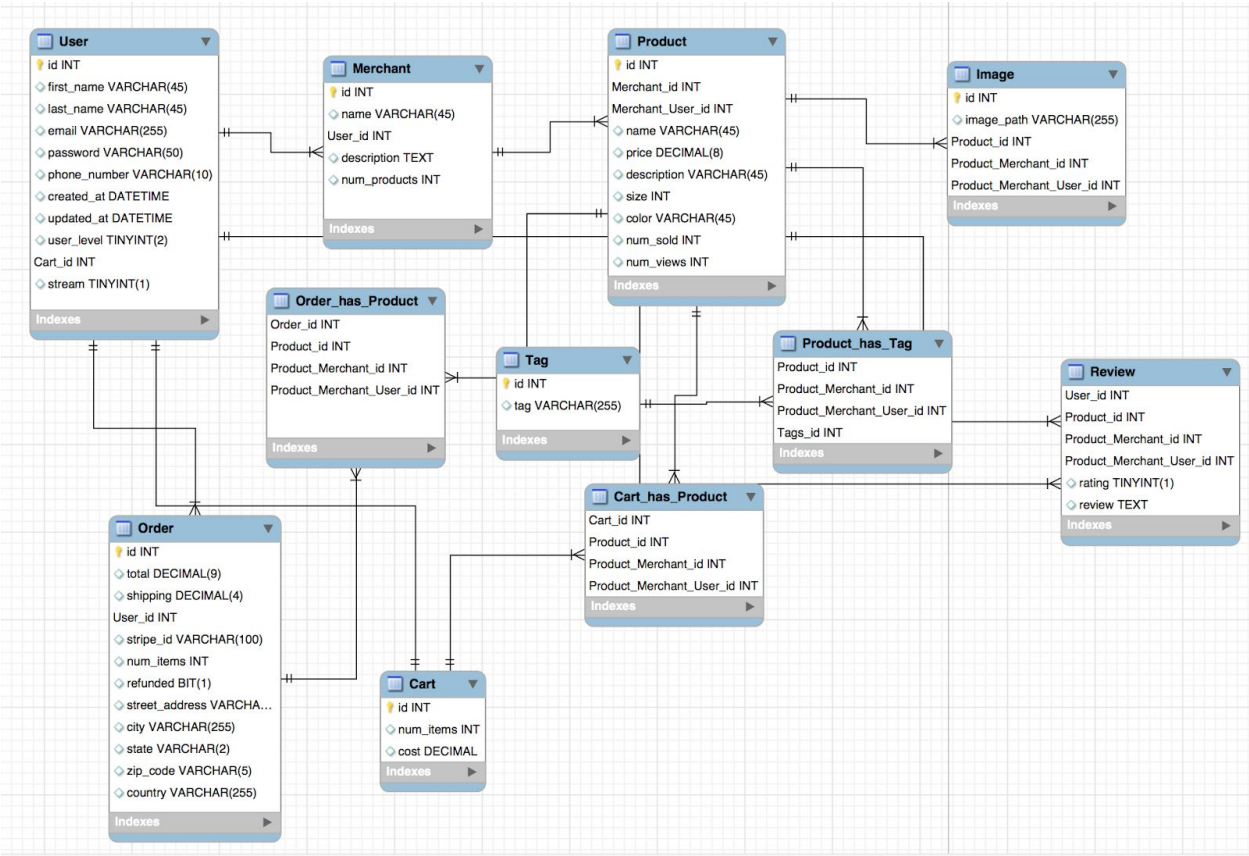
Choice: option A

Justification: Angular has a steep learning curve, however it is very powerful and modular. Angular allows us to separate every part of our project cleanly and easily integrate them together using components. This addresses the question of how we deal with any dependencies for any features that we implement into Congo. Because of this consistency, it's easy to maintain in production as well.

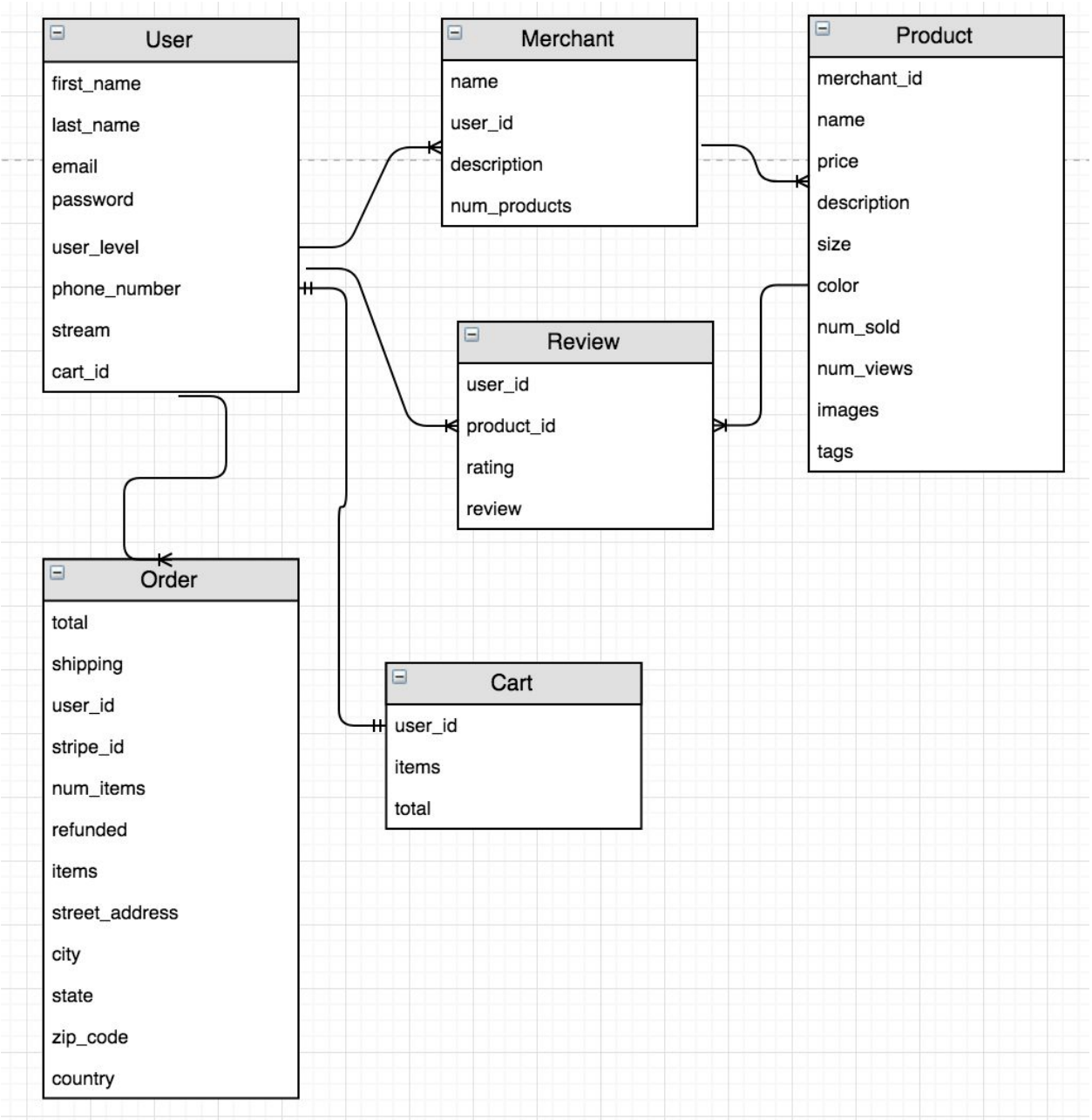
Design Details

Class Design

Database Diagram:


User
id INT
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(255)
password VARCHAR(50)
phone_number VARCHAR(10)
created_at DATETIME
updated_at DATETIME
user_level TINYINT(2)
Cart_id INT
stream TINYINT(1)
Indexes
Merchant
id INT
name VARCHAR(45)
User_id INT
description TEXT
num_products INT
Indexes
Product
id INT
Merchant_id INT
Merchant_User_id INT
name VARCHAR(45)
price DECIMAL(8)
description VARCHAR(45)
size INT
color VARCHAR(45)
num_sold INT
num_views INT
Indexes
Image
id INT
image_path VARCHAR(255)
Product_id INT
Product_Merchant_id INT
Product_Merchant_User_id INT
Indexes
Order_has_Product
Order_id INT
Product_id INT
Product_Merchant_id INT
Product_Merchant_User_id INT
Indexes
Tag
id INT
tag VARCHAR(255)
Indexes
Product_has_Tag
Product_id INT
Product_Merchant_id INT
Product_Merchant_User_id INT
Tags_id INT
Indexes
Review
User_id INT
Product_id INT
Product_Merchant_id INT
Product_Merchant_User_id INT
rating TINYINT(1)
review TEXT
Indexes
Cart_has_Product
Cart_id INT
Product_id INT
Product_Merchant_id INT
Product_Merchant_User_id INT
Indexes
Order
id INT
total DECIMAL(9)
shipping DECIMAL(4)
User_id INT
stripe_id VARCHAR(100)
num_items INT
refunded BIT(1)
street_address VARCHA…
city VARCHAR(255)
state VARCHAR(2)
zip_code VARCHAR(5)
country VARCHAR(255)
Indexes
Cart
id INT
num_items INT
cost DECIMAL
Indexes

Class Diagram:



Description of Classes and Interactions

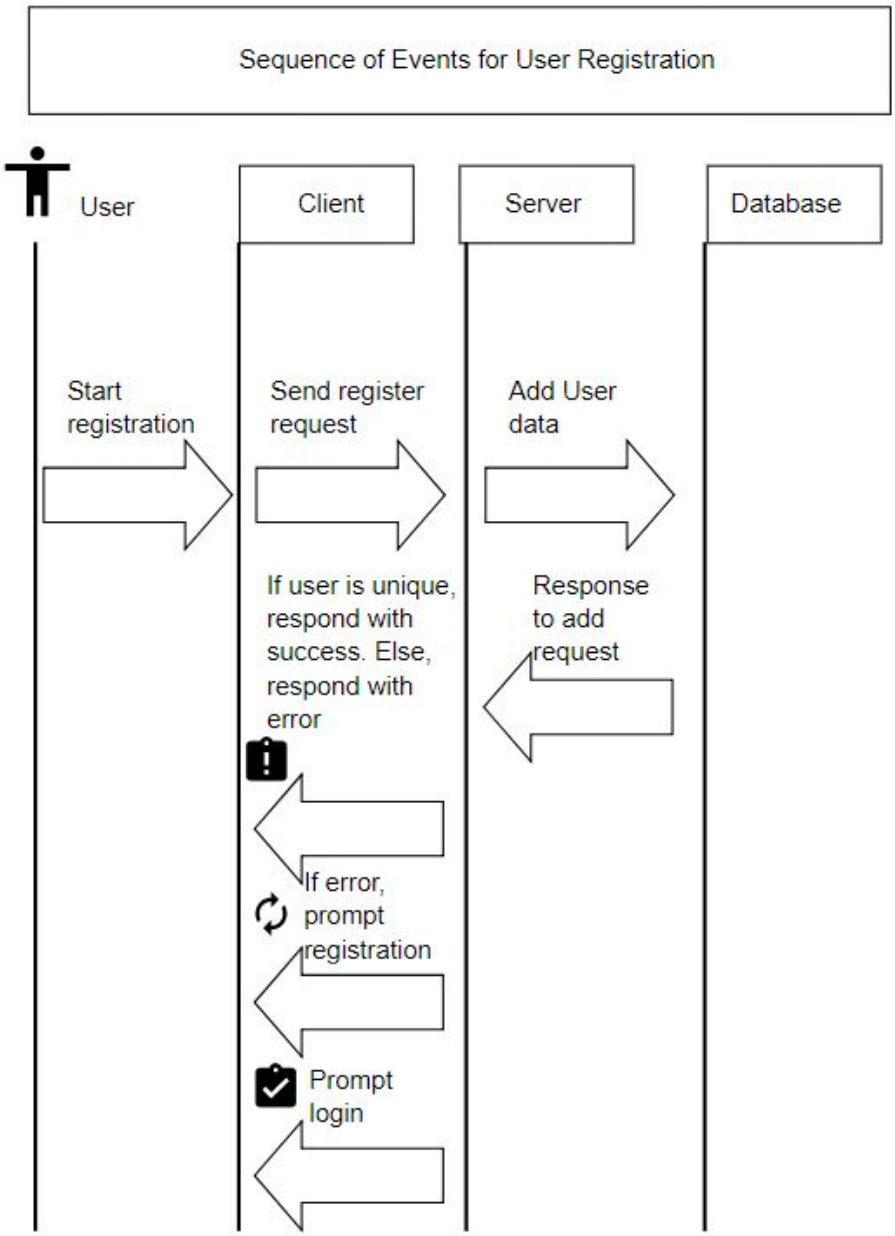
- **Users -** Every client who registers with our site is a user
    - Each user has a first name and last name to personalize the UI with their unique name

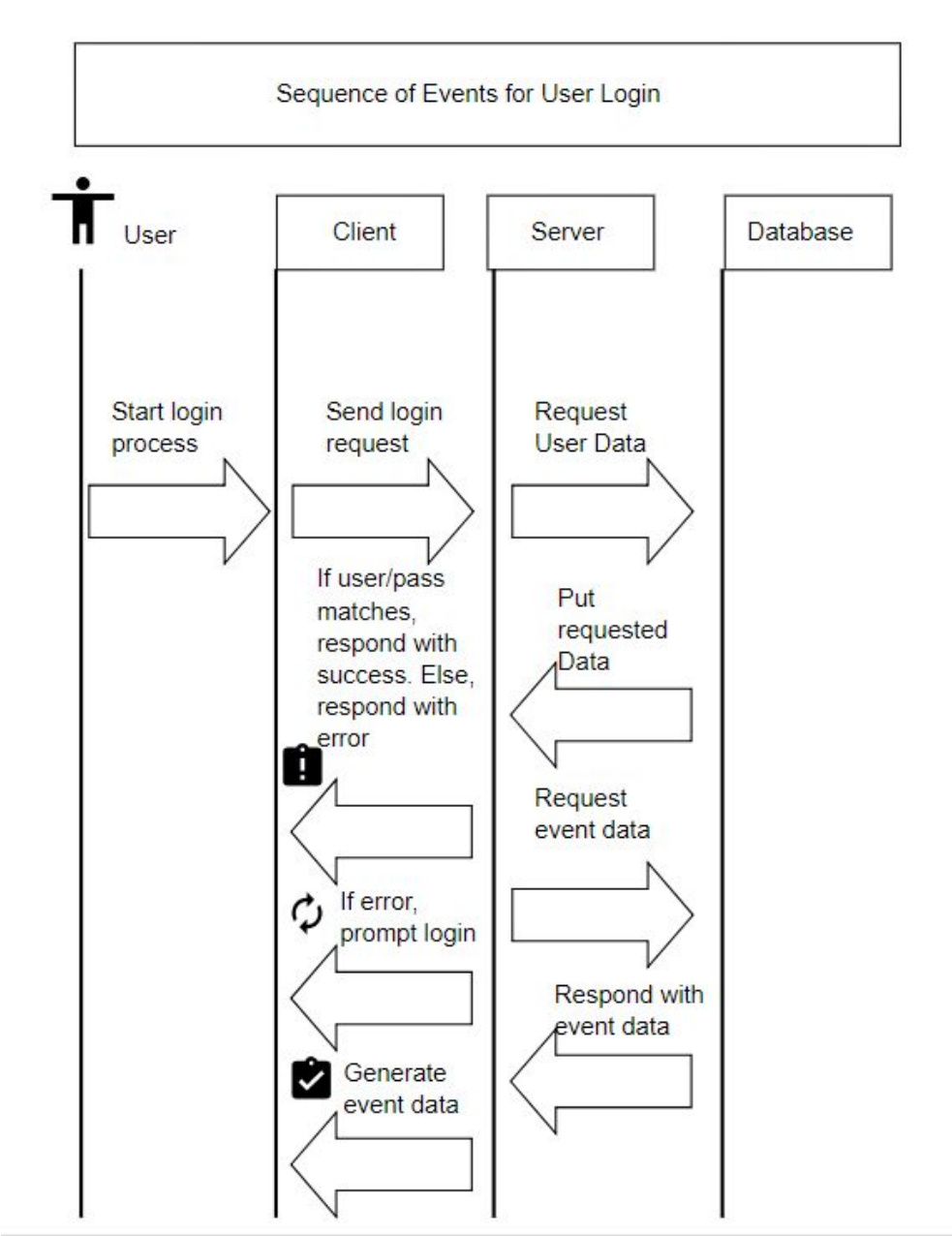
    - Each user has a phone number for contacting if necessary
    - Each user has an email and password for login purposes
    - Each user has a user level, 0 for regular user, 9 for admin, to determine what kind of privileges they will have on the site
    - Each user will contain a created_at tag and updated_at tag to remember these datetimes
    - Each user will have a unique cart_id to show which cart is theirs
    - Each user will have an attribute to signify whether they are a part of our premium subscription service, stream
- **Merchant -** Users can become verified merchants through our verification process, allowing them to sell items
    - Each merchant will have a name for their organization
    - Each merchant will have a user_id to show which user owns the organization
    - Each merchant will have a description to have a way to show customers who they are, and what they are about
    - Each merchant will have an attribute for number of products they are currently selling
- **Product -** Products that a merchant has added to sell
    - Each product has a merchant id, to show who is selling it.
    - Each product has a name, description, and price
    - Each product has an optional size and color field, useful if the product comes in multiple sizes or colors
    - Each product also contains the total number of views, and total number sold to allow merchants to track their statistics and conversion rates.
    - Each product also has images that merchants upload to visually show their product
- **Tag** - Keywords that are related to the product being sold
    - Each tag object contains the tagged word
    - Each tag contains a list of product ids that have this tag
- **Review** - User feedback on a product
    - Each review contains the id of the user who posted it and the id of the product it is for
    - Each review contains a rating out of 5, and a review, a written explanation of their rating,
- **Cart** - The items that are going to be checked out when completing a purchase
    - Each cart contains a list of all the products that are in it
    - Each cart contains the total cost of the transaction, excluding tax and shipping
    - Each cart contains an attribute for the number of items
    - Each cart contains the id of the user it is for
- **Order** - A transaction which has been completed
    - Each order contains the number of items, total cost of the order, and the shipping cost of the order in case of refund
    - Each order contains the id of the user who placed it

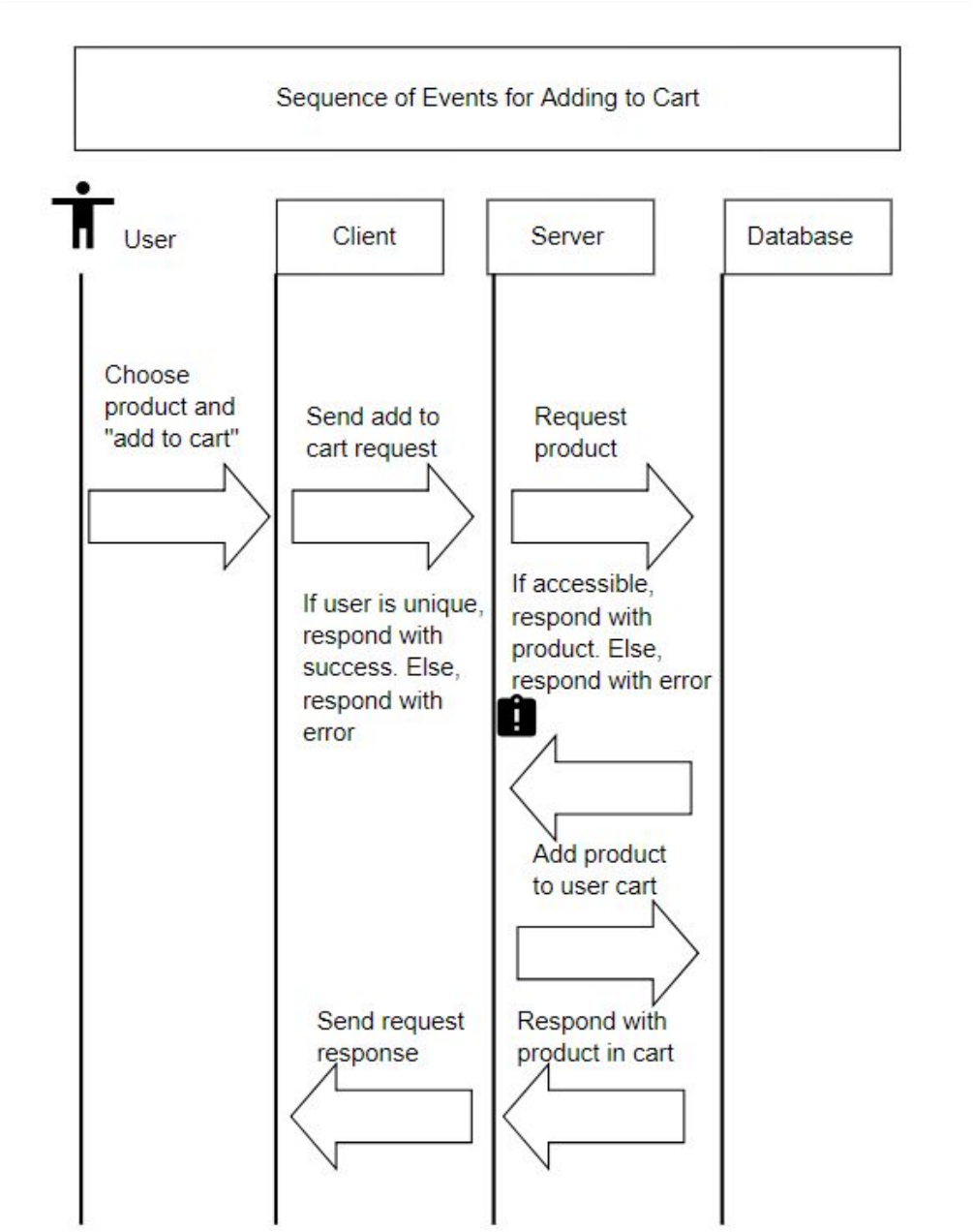
- Each order contains the stripe id of the payment, to refund the user's card if they return
- Each order contains a boolean that keeps track of whether the order has been refunded or not
- Each order contains street address, city, state, country, and zip code of the shipping address

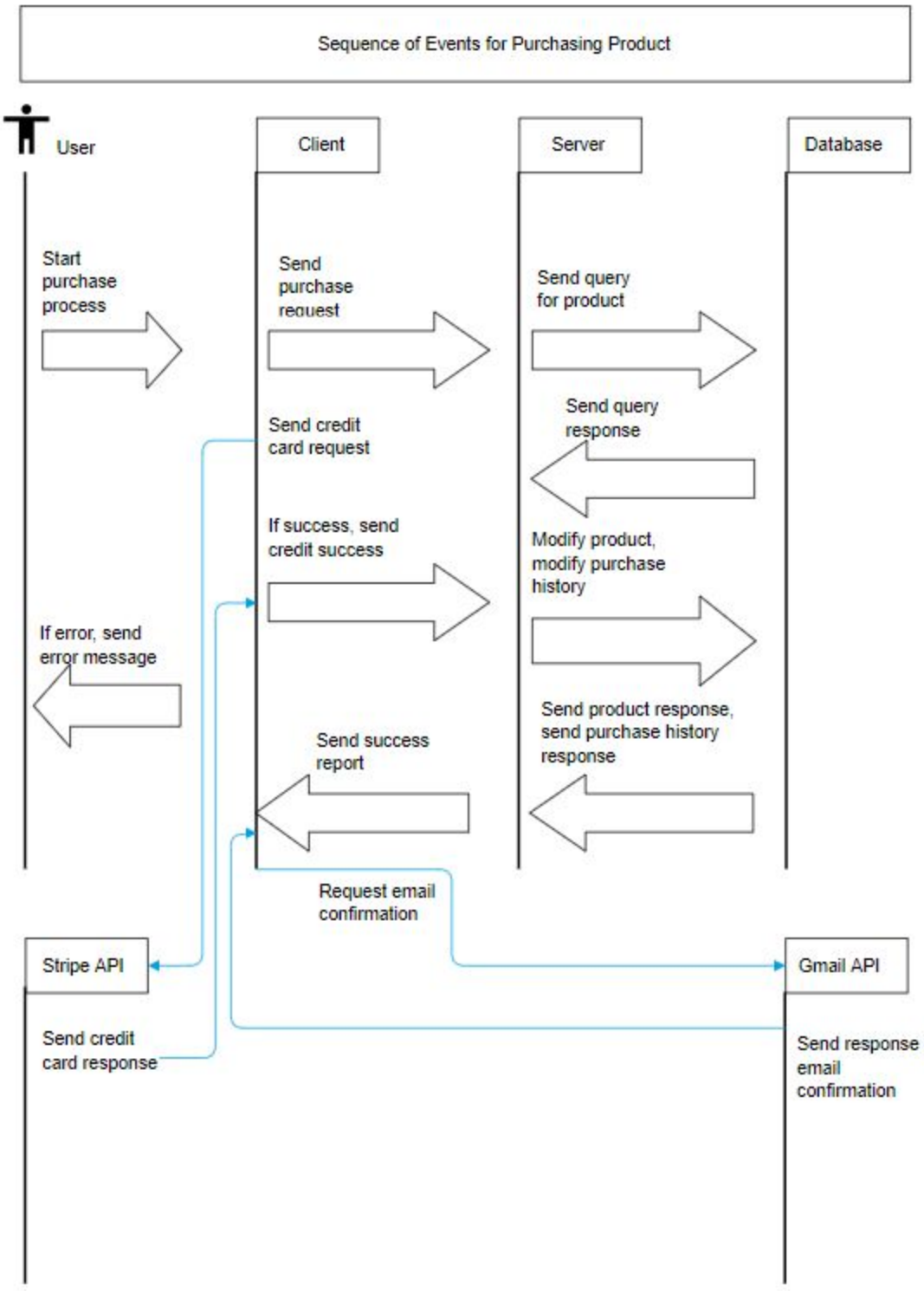
Sequence Design

The sequence diagrams depict four of the processes that our website performs. These processes are as follows: user registration, user login, adding to cart, and purchasing a product. These diagrams show the connection between a user and the client, the client and the server, the server to the database, and the client to the stripe and Gmail api's. The connection between the user and the client will show the visual representation of the marketplace. When the client sends a request, it will be sent to and satisfied by the server. The server retrieves, adds, and modifies the data from the database and sends it back to the client. Another connection is between the client and the stripe api/Gmail api. These are used for credit card purchases and email confirmation for successful purchases, respectively. The client will process the data and update the website accordingly.


Sequence of Events for User Registration
User
Client
Server
Database
Start registration
Send register request
Add User data
If user is unique, respond with success. Else, respond with error
Response to add request
If error, prompt registration
Prompt login


Sequence of Events for User Login
User
Client
Server
Database
Start login process
Send login request
Request User Data
If user/pass matches, respond with success. Else, respond with error
Put requested Data
Request event data
If error, prompt login
Respond with event data
Generate event data


Sequence of Events for Adding to Cart
User
Client
Server
Database
Choose product and "add to cart"
Send add to cart request
Request product
If user is unique, respond with success. Else, respond with error
If accessible, respond with product. Else, respond with error
Add product to user cart
Send request response
Respond with product in cart


Sequence of Events for Purchasing Product
User
Client
Server
Database
Start purchase process
Send purchase request
Send query for product
Send query response
Send credit card request
If success, send credit success
Modify product, modify purchase history
If error, send error message
Send product response, send purchase history response
Send success report
Request email confirmation
Stripe API
Gmail API
Send credit card response
Send response email confirmation

Navigation Flow Map

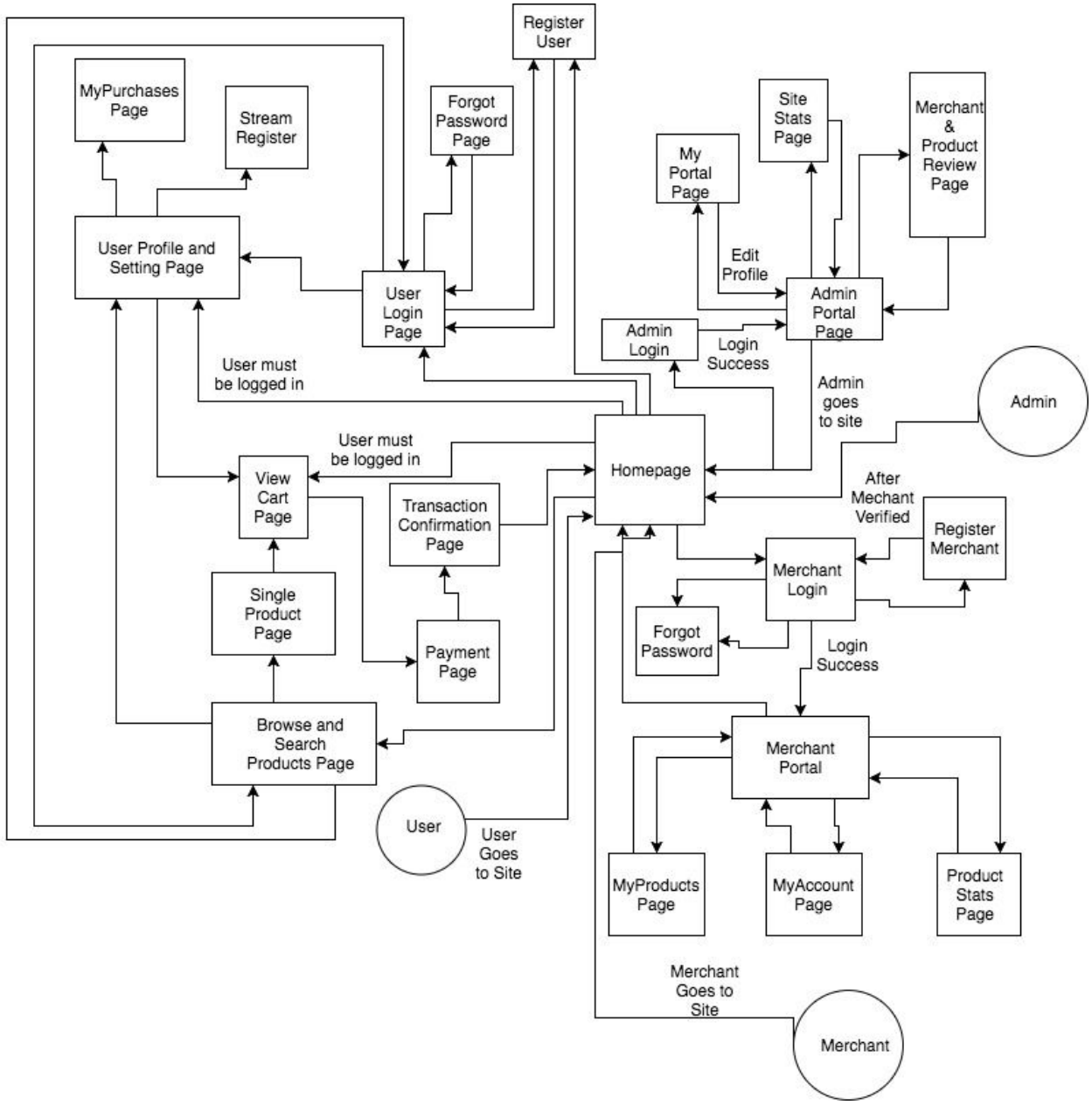
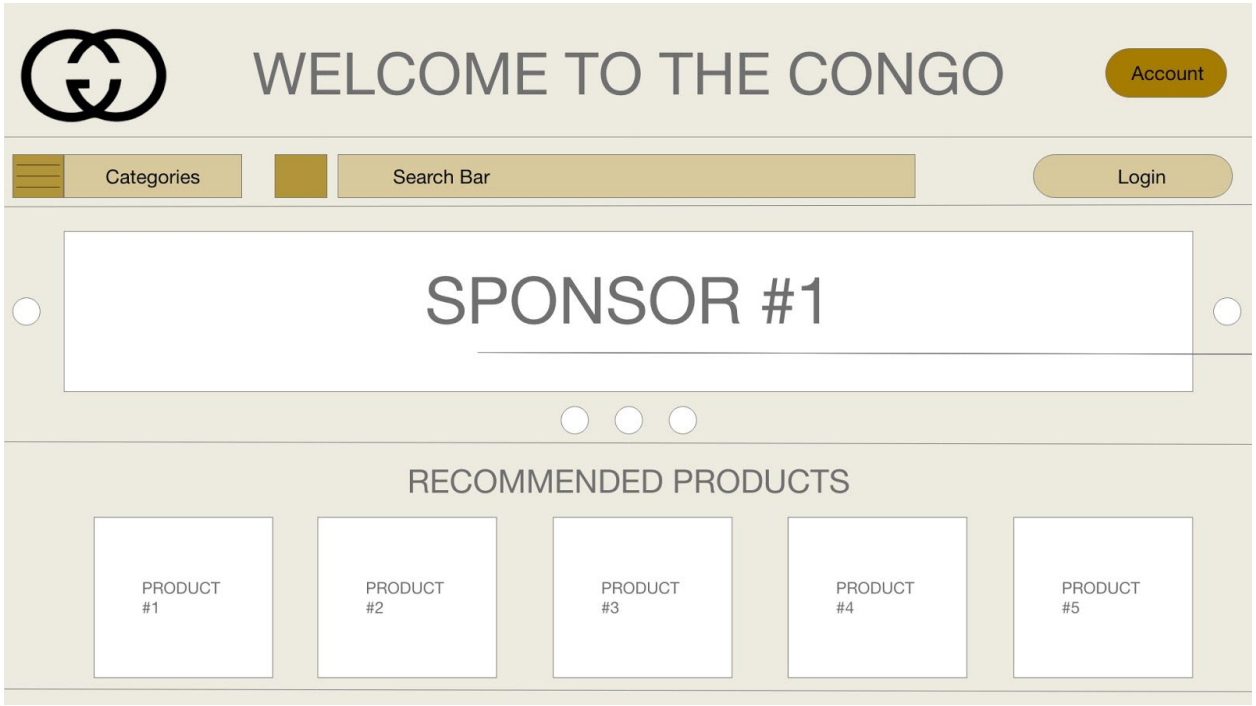


Diagram shows the ease of access. The homepage will be the main point of access to most of the pages. The merchant, admin, and user access different sectors of Congo as you can see from the flow chart. However, the user's access to the site will be the most complicated to implement.

UI Mockup

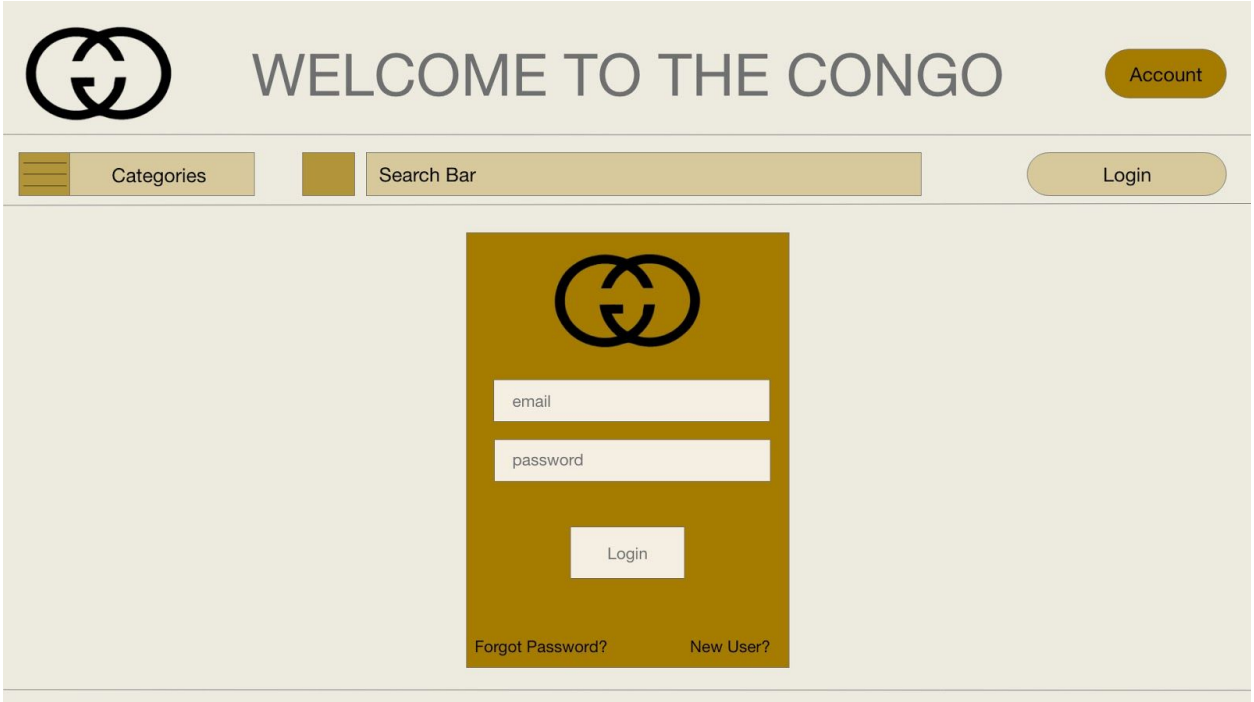
We designed the header bar of the website to be available through every single page. The header bar contains quick access to the user's profile page, and is able to search products regardless of what page you are on. The UI is clean, well-organized simple, and efficient. Our logo will also be a button that will directly take the user to the homepage of your site. By having the header on every single page, we give the opportunity for the user to search for more products without returning to the homepage.

Mockup #1: Homepage



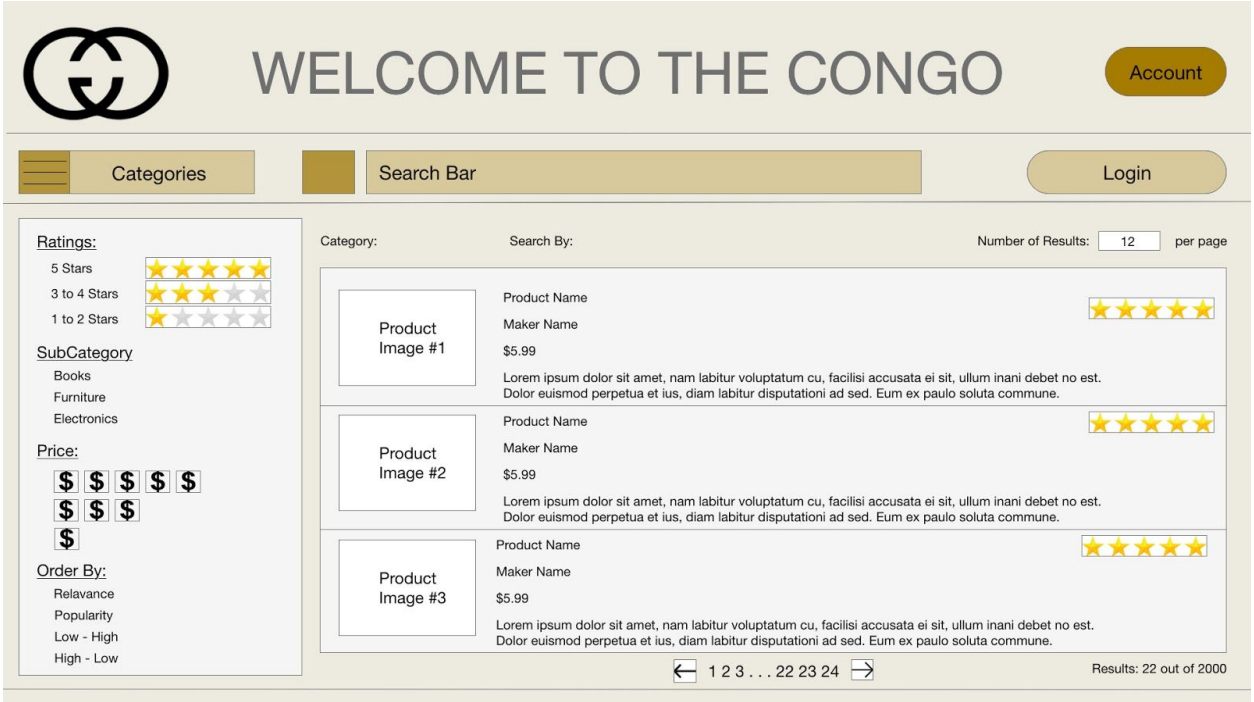
This is the homepage and the first thing our users will view when directed to our site. As a normal e-commerce site, we have a consistent yet simple header where users can sign in and search products in our database. We also have a sponsor section, to allow merchants who want to promote their products on our homepage. Below that we will have a series of sections dependent on the user such as recommended products, related products, browsing history, and related sponsored products based on user profile.

Mockup #2: General Login Page



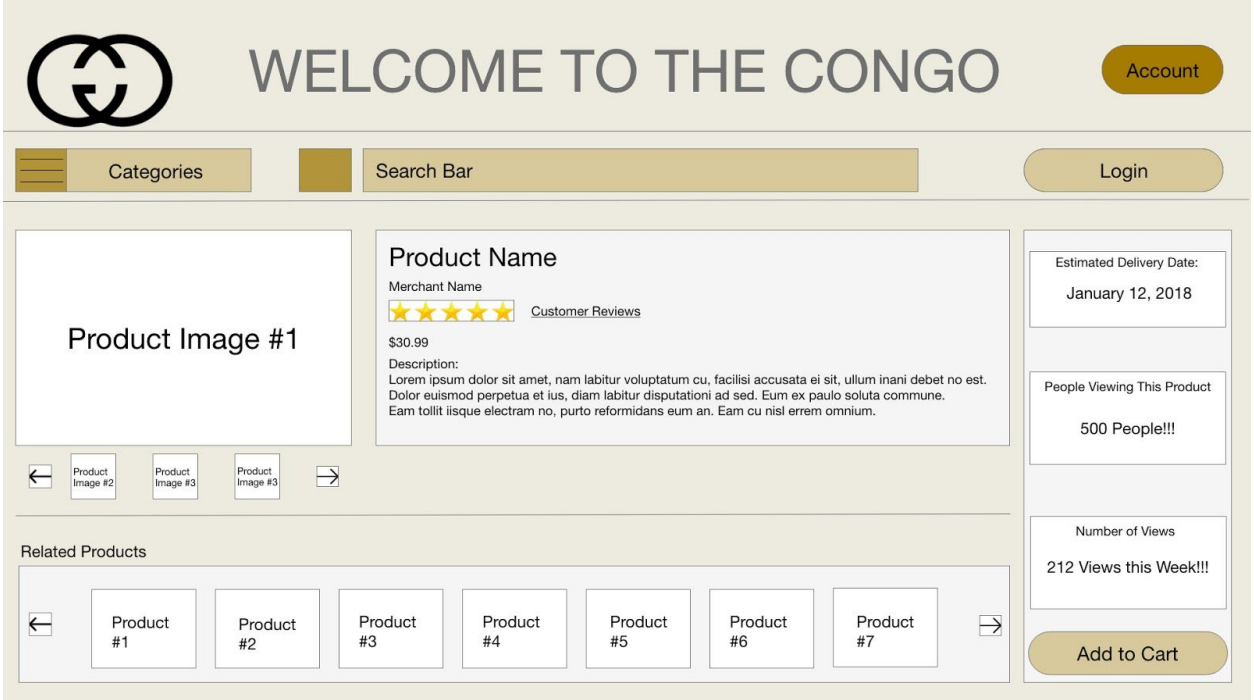
This is the login page for our users. This will be the main login page for our Congo members. Our merchants and admins will have a more tailored login page; however, this will be the default and base layout. We wanted to keep it simple and functional, easy to understand, and look modern to our users.

Mockup #3: Search Page



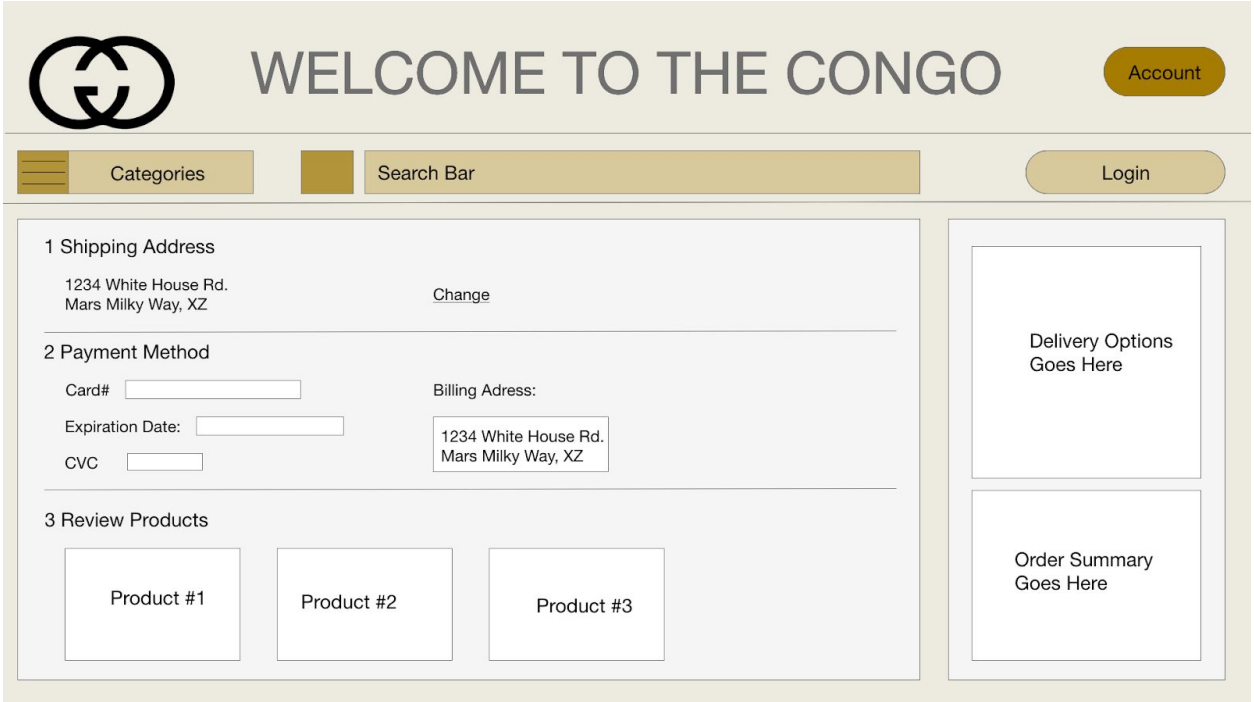
This will be the basic format for when a user searches a product on our website. As you can see this page is very functional and easily understandable by the different sections. We have a filtering section, where users can filter their search, so they can find the product they want. We also give the ratings and a brief description for each product displayed on the section. This gives the user more information to make the right decision.

Mockup #4: Individual Product Page



This is the default page for when the user clicks on an individual product from their search. Along with showing the basic information of the product, we also give the user all the related products based on their search and the products on this page. This allows the user to browse more on our site and make a more intellectual decision. On the right side, we have added a new feature. We will be informing the user how many other users are view this product right now and how many have viewed it this week. With all this information on this page, we want to help the user to make the best decision they can.

Mockup #4 : Checkout Page



This is the Payment Page, where the customer will review their order and products. They will input all the necessary information to process the order and choose their delivery option. We hope the simplicity and ease of use for the page attracts the user. We want to make sure that the overall process from entering our site until processing their order is smooth, efficient, and customer oriented.