ENSF 607 / 608 Project Proposal / Report

Geer Ma, Mike Ebrahimi, Kendall Reed

Full-Stack E-Commerce Web Application: "Active Tech Style"

Project abstract

This project is an E-commerce application. Enormous multinational corporations such as Amazon and Shopify signify that E-commerce is an area that is extremely valuable and useful to consumers. The ability to quickly order online products and goods from any convenient location is very useful. Our project aims to take inspiration from those sites to create an application that allows users to view and purchase items from a variety of categories.

Project scenario and goals

The user interface will be a React.js web application that is deployed to a service such as AWS or another cloud-based deployment platform. We will need multiple pages to handle the different categories and different functionalities. To simplify our project, we will only make the user interface for the CUSTOMER (we will not implement the ADMIN side of the application). Ideally, we would want our E-commerce application to be fast and be able to handle a huge amount of items and categories - however, for our project, we will aim for only 3 categories (so that each person can do one).

Design strategy

We aim to have a client-server design for our application. The front-end client will be done via React (along with HTML/CSS/JavaScript). The back-end client will be done via MySQL as the database, and Java along with Springboot as the framework. They will communicate via API requests. The React will fetch a GET request from the API endpoint when it wants to receive information such as items. When purchasing items, it will likely need to submit a POST request to the MySQL database.

Design unknowns/risks

Since we are trying to create a new database schema implementation instead of using the default Student Course Registration, that would be the first issue to tackle. We would likely need to get some feedback from the TAs or course instructor if we get stuck.

Some difficulties that we envision having to deal with could be trying to relate the products with the cart, and adding items. We also would need to look up how to deal with images in the database as well mapping out every item from the front-end. The search and filter functionality may be something that would also be difficult to implement, and we may decide to have a primitive search functionality (or not have one at all and only browse by category).

In terms of the user interface, we have one member who is comfortable with front-end design and two members who are learning it. The members who have no experience with front-end development might have to learn many new things, but the member who is comfortable with it can teach and mentor, and set-up the front-end side of the project.

Implementation plan and schedule

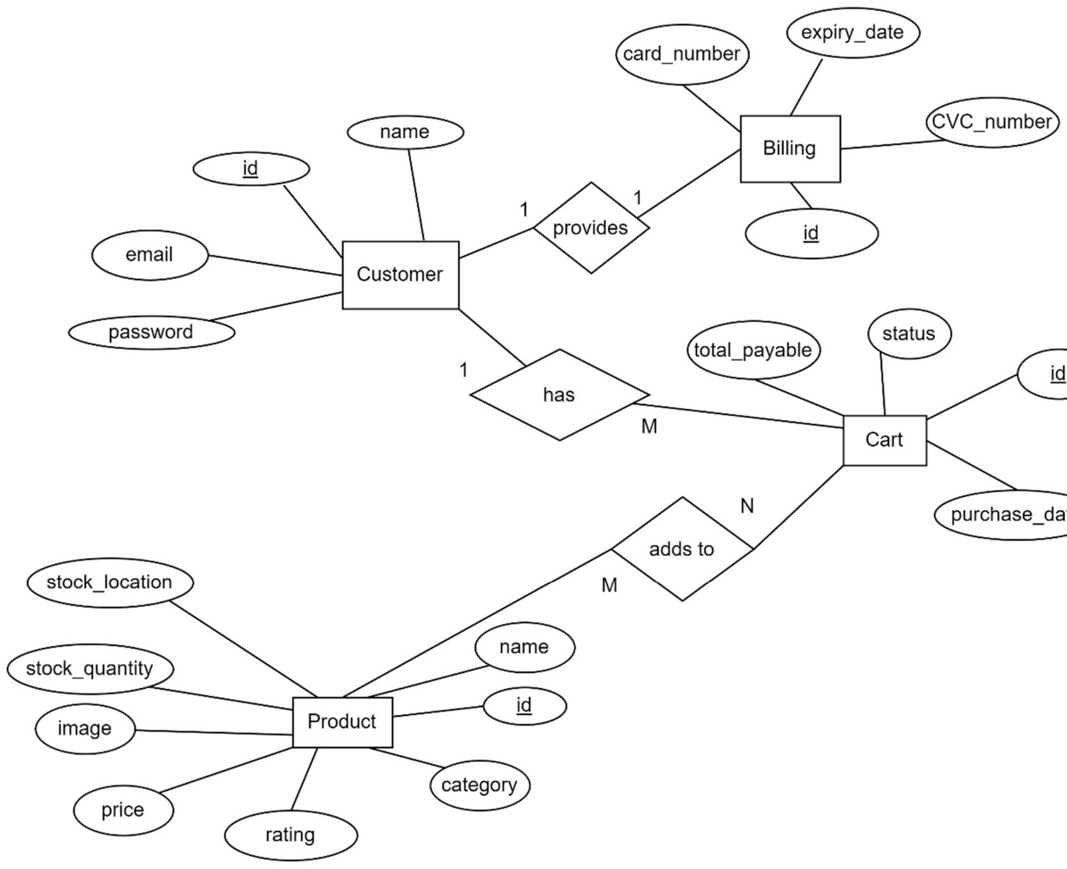
First of all, we would like to adhere to the ENSF 608 submission deadlines for each deliverable in order to receive feedback on our Database progress. After finishing the Proposal and ER Diagram by Nov 18th, we hope to finish The relational model by Nov 25th, and then the rest of the deliverables by the final due date of Dec 2.

We hope to start and finalize a good amount of the database by the end of the November 27th weekend. We will start a draft of the front-end before this date, but the majority of the front-end design would most likely happen over the week of November 28 to December 1. The back-end and front-end can be developed separately, then connected via API requests when both are more developed.

Evaluation

We can evaluate success by how much we actually implemented versus what we plan to implement. If we were able to implement most, if not all, of the features that we plan for (ex. Functional React front-end that can see a list of all items in a category, search and filter, add to cart, and purchase), then we will say that we are very successful. However, we anticipate that there will be areas that we may not be able to implement due to time constraints since we only have 2 weeks to finish this project. However, as long as we attempt as much as we can, learn from our mistakes, and write about our experience in the final report it will not be considered a failure. The tradeoff is that we want to prioritize the basic functionality of our project such as the item and category database. Having a polished UI would be great, but it is not a huge priority for our group.

ER Diagram



- **Assumptions:**

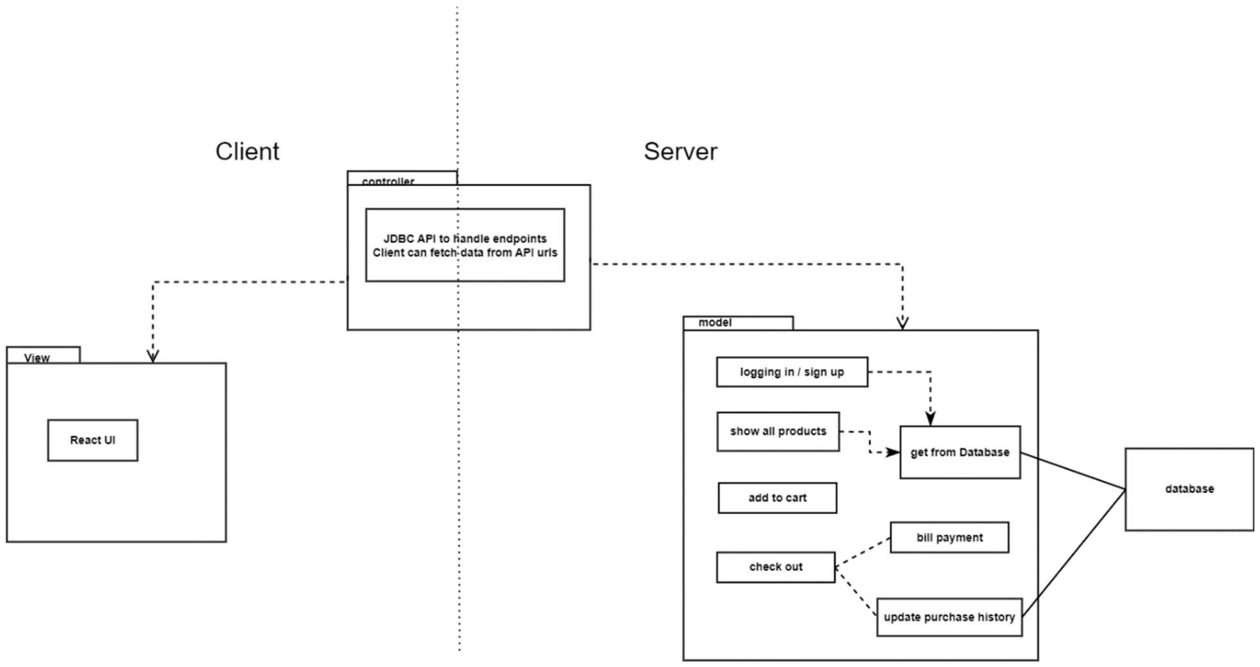
  1. user can only add one item of each product to the cart, meaning that cannot have multiple items of same product in the cart.
  2. cart could supposedly be used for keeping track of customers purchase history as well as the wish list by assigning a proper “status” attribute.

3. relation between customer and billing is 1-1 however both can exist without the need to be assigned to the other one (a card can exist without the customer and vice versa). However, the frontend prevents submitting any order for a customer without having a billing assigned to it.
4. all users must be registered to be able to checkout.

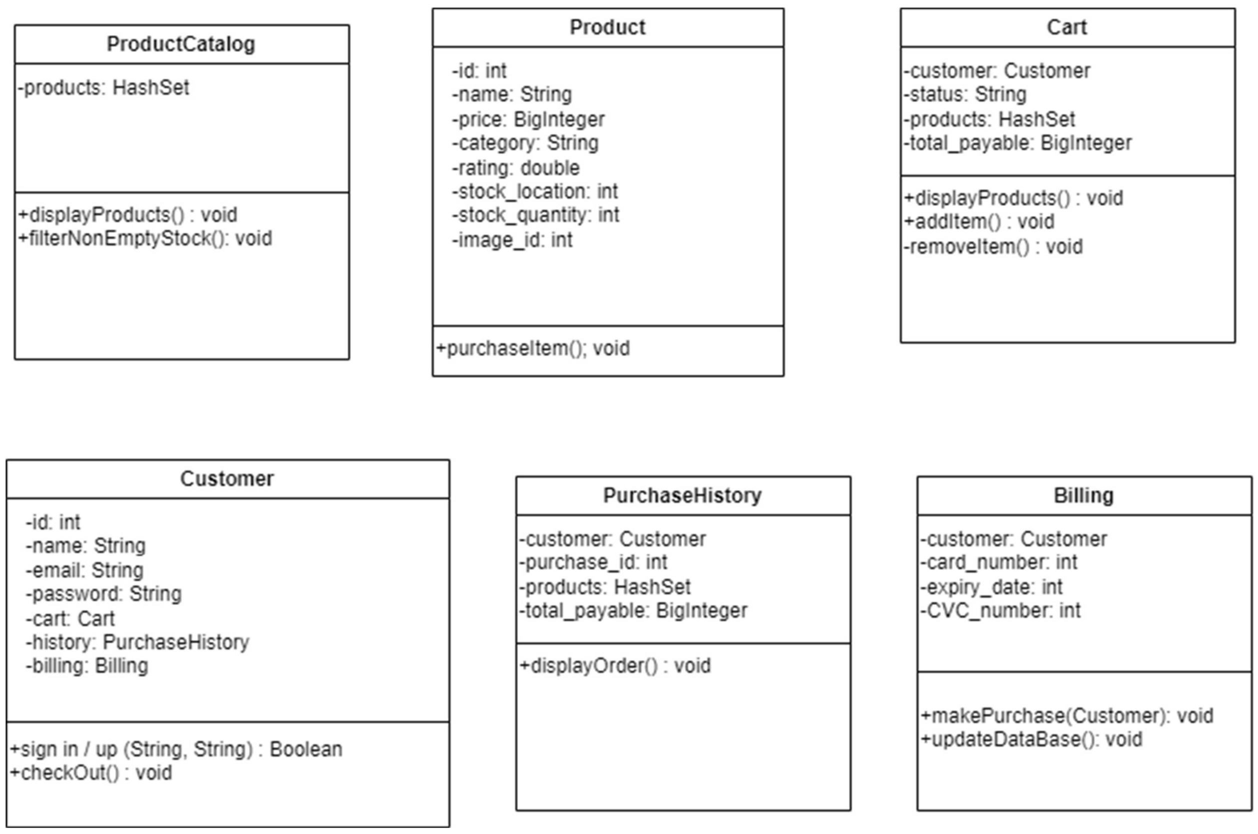
Note: During the course of design and implementation of the project, we realized that a better schema could be used for this project, therefore, we changed the schema as well as the relational model. As a result, the implemented design is a bit different than what was initially proposed.

Note that CART is essentially an “order_history”. A TA from ENSF 607 suggested that we handle the cart feature on the frontend. So essentially the frontend will have a list of products that will be sent to the backend when the customer goes to purchase. Then, it will create a cart but have the status automatically set to “completed”. This means that it is a completed cart order. The alternative is that the status is “pending”, which is more similar to a wishlist (not implemented for our project).

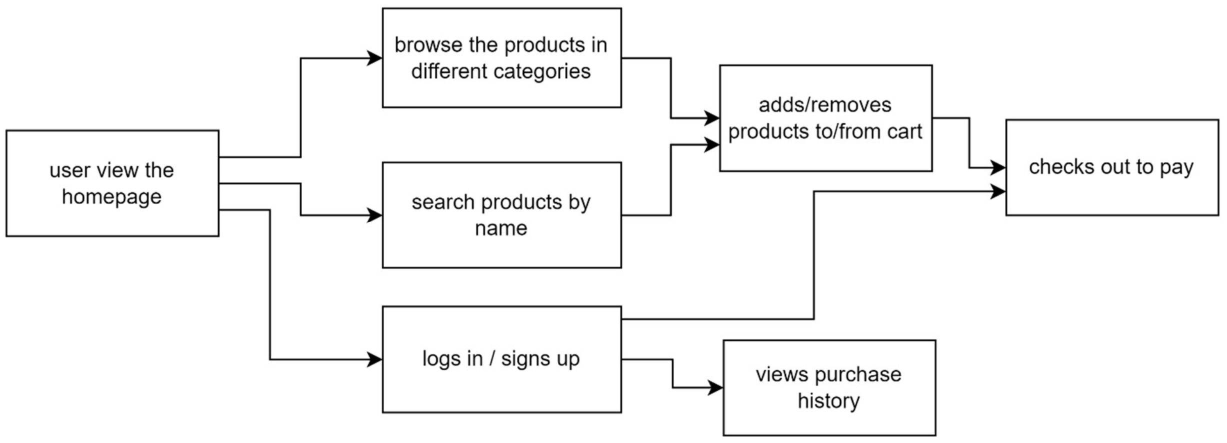
MVC Design Architecture



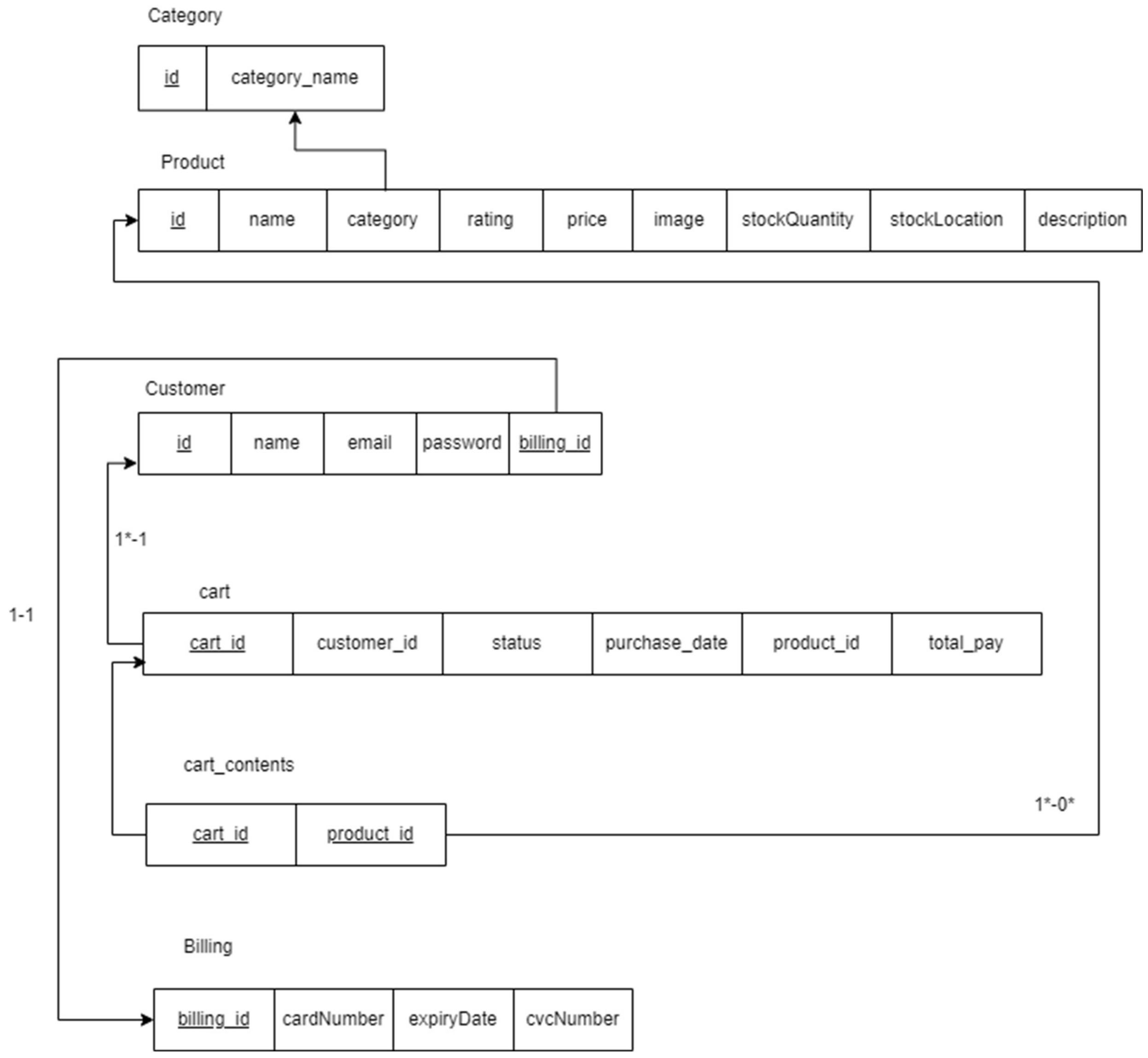
Classes



Functional Mode



ENSF 607 / 608 Relational Model


Category
id
category_name
Product
id
name
category
rating
price
image
stockQuantity
stockLocation
description
Customer
id
name
email
password
billing_id
1*-1
1-1
cart
cart_id
customer_id
status
purchase_date
product_id
total_pay
cart_contents
cart_id
product_id
1*-0*
Billing
billing_id
cardNumber
expiryDate
cvcNumber