

# **Electronic Vendor Database**

Design and Implementation of a Database for Electronic Vendors

**Joel Ponder**  
A20316167

**Oscar Sanchez**  
A20374360

**Nicholas Tating**  
A20452101

**Illinois Institute of Technology**  
**CS425-Database Organization**  
**Professor: Yuan Hong**

## Project Description

---

Our project is an implementation of a relational database for an electronic vendor. The objective was to design the database to be similar to that of Best Buy, Circuit City, etc. We have a hypothetical company that has assigned our team to redesign a major part of the database that underlies the company operations. The first task of our project was to create an entity relationship diagram (ERD) that stored data for the major components of the electronic vendor database. We created an ERD to support a number of functionalities including:

- Grouping products into categories
- Coordinating billing accounts for companies
- Providing tracking IDs to shipping companies and customers
- Updating inventory accordingly, including automating orders
- Storing payment information for customers that are infrequent

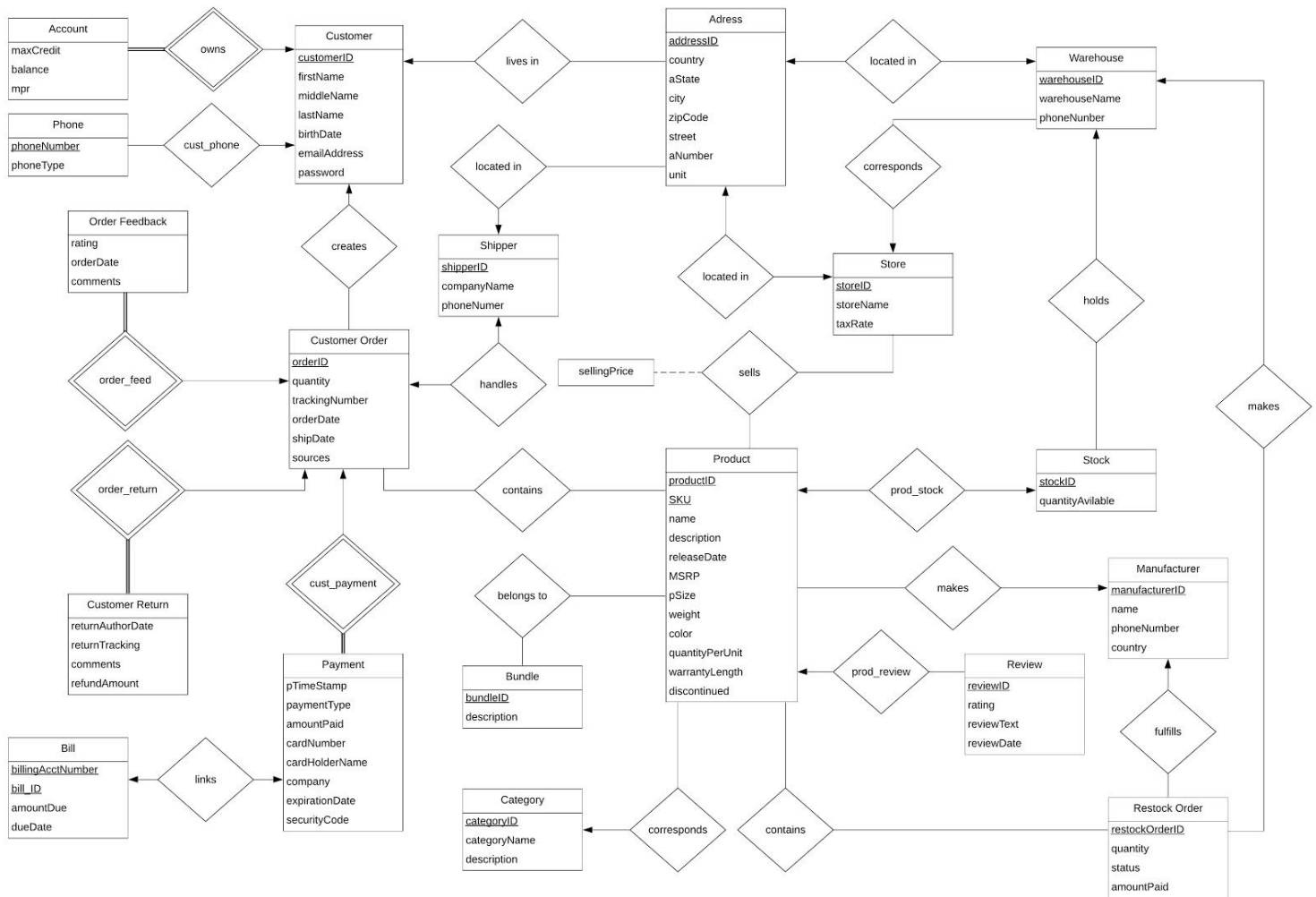
We decided to add some additional features to the database by allowing users to make returns, leave feedback for their ordering experience, and write reviews for products they purchased. Note that these are optional features which means they do not add any storage overhead for our database. Our ERD consists of 25 different entities representing different components of the database such as payment information or customer orders and relationship sets between entities. Billing accounts were created for frequent customers to make their purchases with the credit available in their account. Information for each retail or online store is stored separately to allow stores to vary the price of their products sold. The amount of stock available for each product at different stores is also stored in our database. Database administrators for this electronic vendor have the ability to store more detailed information as needed by altering the entities we have created. The ERD corresponding to our database is shown on the following page.

In addition to the above requirements, we created a desktop application that supports the following actions:

- Account registration for frequent customers
- Modification of account information
- View purchase history and stock information
- Verification of available credit

## ER Diagram

### CS425 FINAL PROJECT: ELECTRONIC VENDOR ERD

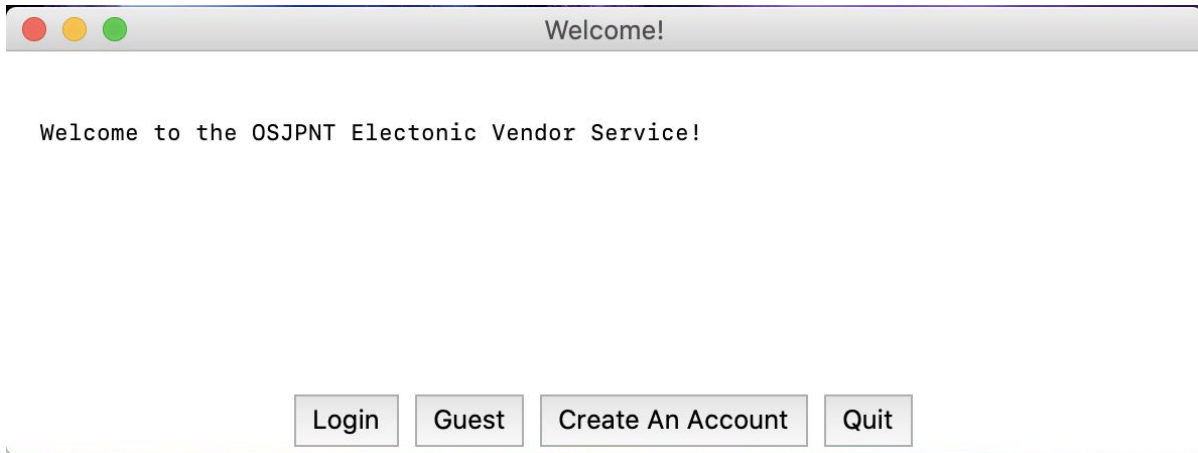


\*\*\*Please review the PDF file attached to our submission to obtain a more detailed view of the various entities and relationship sets\*\*\*

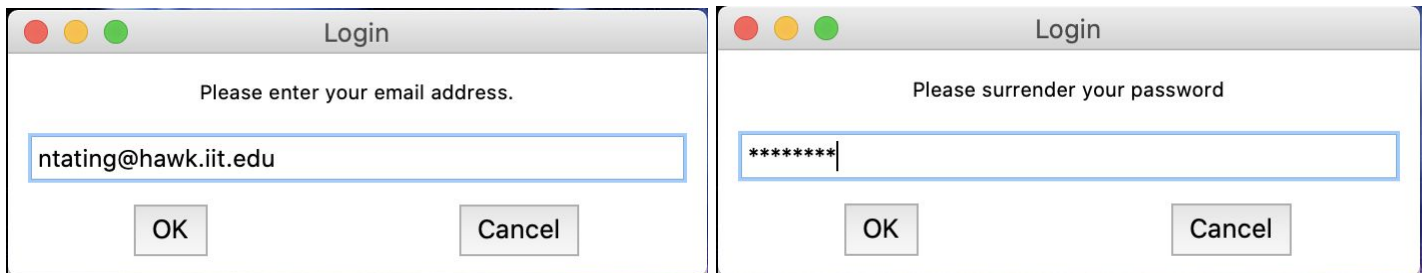
## Application Images

---

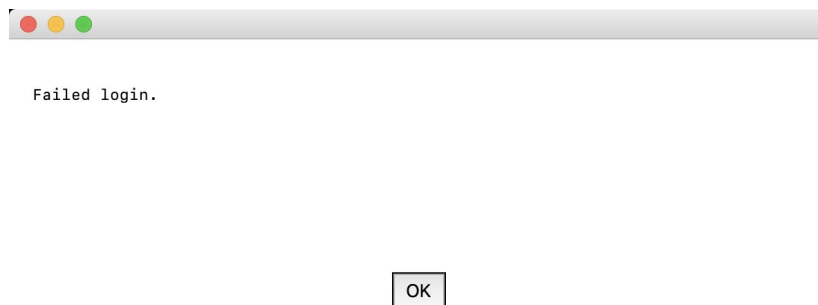
The application boots when the file `customer.py` is executed in shell with `python3`. The application uses the following libraries: `easygui`, `mysql.connector`, `datetime`, `sys`. Upon execution, the main menu will pop up.



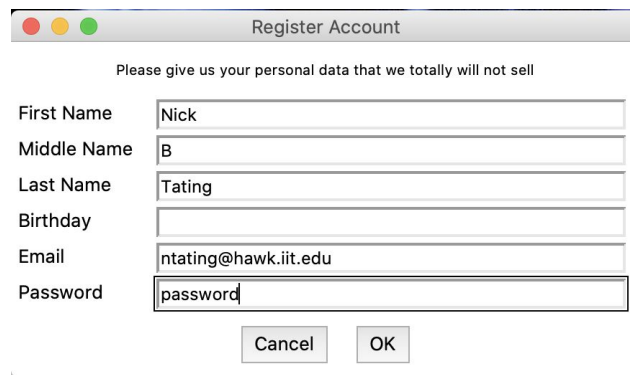
The main menu gives the user four options. The user can either login with their credentials, login as a guest, create an account, or quit. Quit is the simplest, as it simply closes the program. We will demonstrate the actions of each button throughout this section, starting with the Login button.



The user may populate the boxes with their login credentials, as I have done above. Observe that the password is masked in order to protect the user's data privacy. Once the user inputs their login credentials, our mysql queries determine if the user has input a valid login credential.

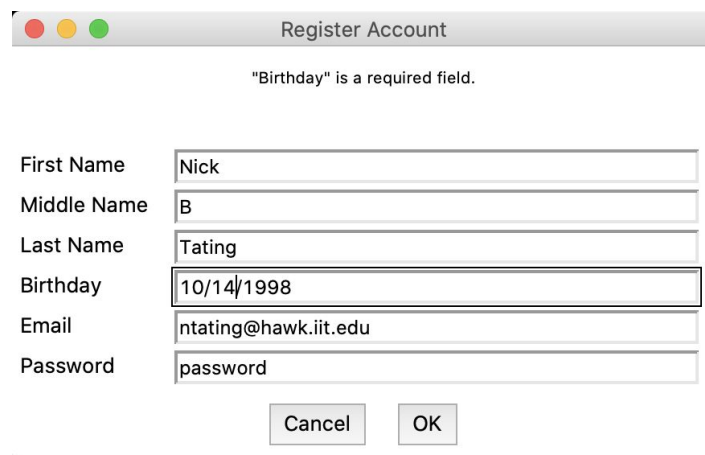


Oh no! I have failed to log in. But I expected that, seeing that I've never actually created an account with our program. But let's do that now! Instead of hitting "Login" on the main screen, let's do "Create An Account."



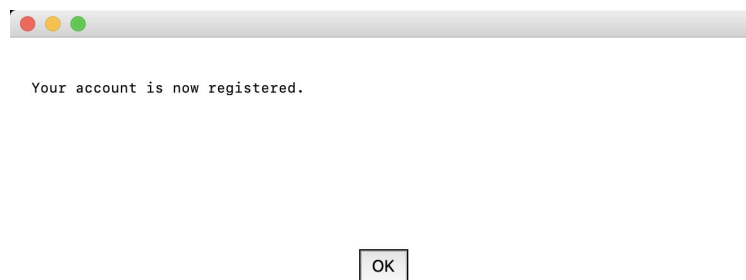
A macOS-style dialog box titled "Register Account". It contains a message: "Please give us your personal data that we totally will not sell". Below this are six text input fields: "First Name" (filled with "Nick"), "Middle Name" (filled with "B"), "Last Name" (filled with "Tating"), "Birthday" (empty), "Email" (filled with "ntating@hawk.iit.edu"), and "Password" (filled with "password"). At the bottom are "Cancel" and "OK" buttons.

The user is then prompted to input a set of data that is used in account creation. If the user chooses to omit information, they will be repeatedly be prompted to fill in all of the boxes, that way we can sell their data and make a ton of money. Everyone wants to know what your birthday is because certain signs have a higher tendency to impulse buy, and thus we should market stronger towards them and exploit their birthday.



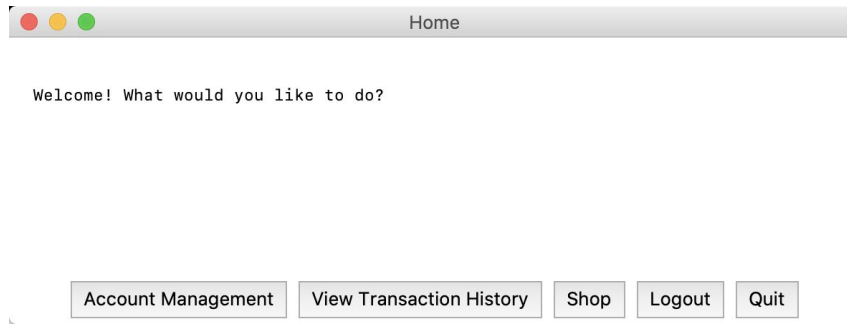
The same "Register Account" dialog box, but with an error message at the top: "\"Birthday\" is a required field." The "Birthday" field is now filled with "10/14/1998". All other fields and buttons remain the same.

Nice try, libras. We are coming after your money and lack of self control. (I speak from experience since I have a tendency to impulse buy and I'm a libra, but quite frankly I don't follow signs. But some marketers do!).

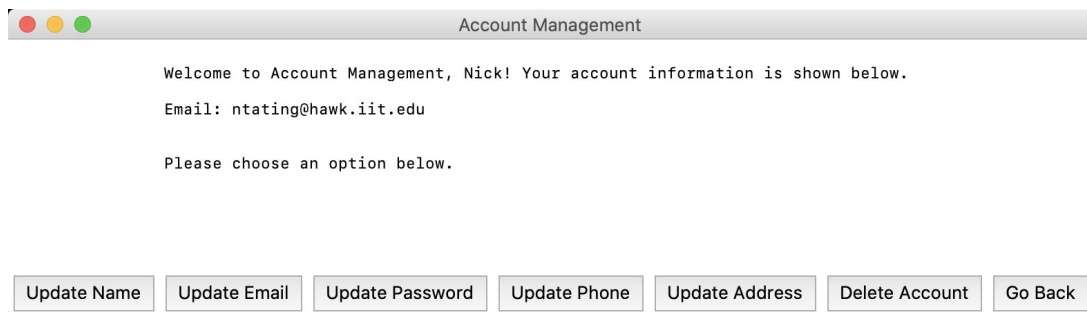


A simple macOS-style dialog box with a single line of text: "Your account is now registered." At the bottom is an "OK" button.

When the user's account has been registered, they will see a confirmation pop-up box just to verify that anything actually happened. Then the user will automatically be logged in and brought to the next menu page.



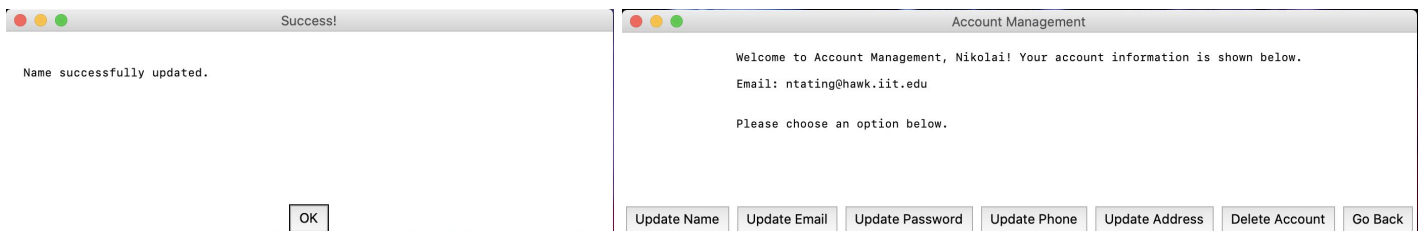
Once the user is logged in, they have a few options as shown above. Quit will obviously close the application. Logout will log the user out, return to the main menu, and allow an opportunity for another client to log in with their credentials or as a guest. Let's look at some account managing options.



As you can see, there are quite a few options in this section of the program. Fear not, we will run through it all!



The update name button prompts the user to input their new names just in case they need to recreate themselves. I will reidentify myself as an East Slavic version of my name.



We were able to successfully change the name, as reflected in both the pop up box on the left above, and the name listed in the welcome greeting in the Account Management tab.

Account Management

Welcome to Account Management, Nikolai! Your account information is shown below.

Email: ntating@gmail.com

Please choose an option below.

Update Name Update Email Update Password Update Phone Update Address Delete Account Go Back

A similar process goes for the email address and password. In the above image, it has been demonstrated that we were able to successfully change the email address corresponding to the account. The password is not so obvious because who in the right mind would display your password out in the open like that?

New Phone

Please fill out the form below.

Phone Number 6502959418

Phone Type Cell

Cancel OK

Now we move on to updating the phone, which now we are prompted to add a new phone and the corresponding type. Please do not blow up my notifications. Just kidding, that's an old number.

Account Management

Welcome to Account Management, Nikolai! Your account information is shown below.

Email: ntating@gmail.com  
Work: 4157168571  
Cell: 6502959418  
Home: 6503597937

Please choose an option below.

Update Name Update Email Update Password Update Phone Update Address Delete Account Go Back

Our program can hold as many phone numbers as the user wants to put in. As shown in the above image, this account has three different phone numbers. Now we add in an old undergraduate mailing address of mine.

Register Account

Please give us your personal data that we totally will not sell

Country USA

State CA

City Berkeley

Zip Code

Street Hilgard Ave

Street Number 2641

Unit

Cancel OK

Register Account

"Zip Code" is a required field.

Country USA

State CA

City Berkeley

Zip Code

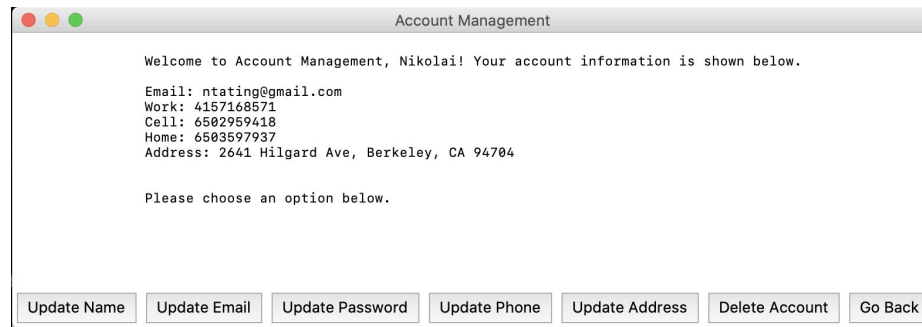
Street Hilgard Ave

Street Number 2641

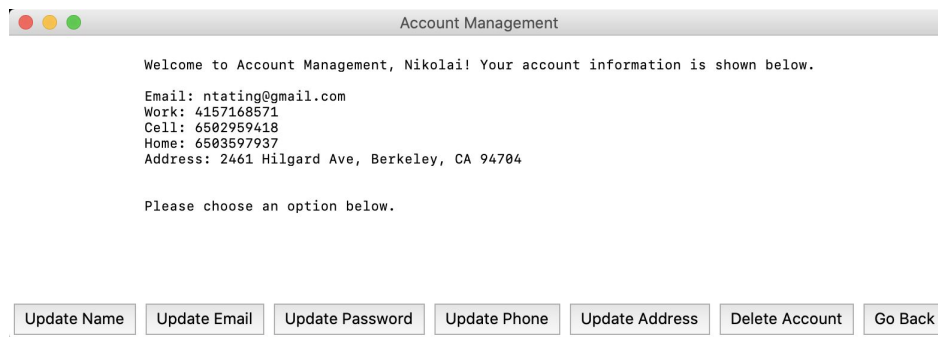
Unit

Cancel OK

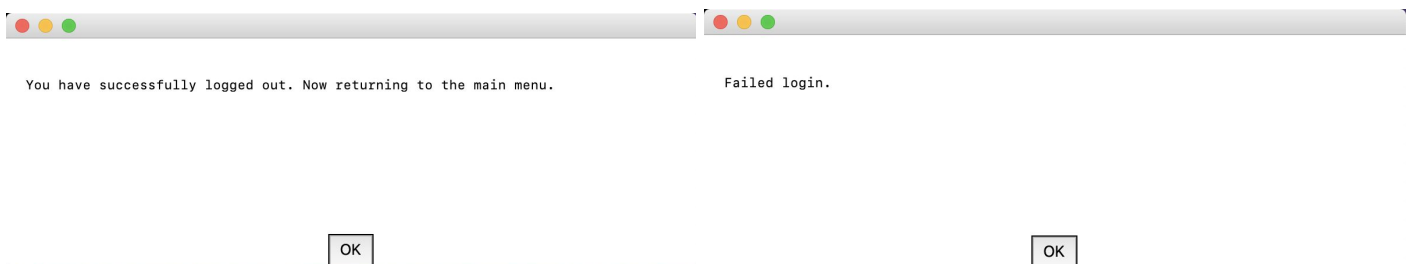
When adding a new address, the user must fill every box except for the unit box since that doesn't apply to everyone. Notice in the image on the right in the previous set of pictures, the program prompts the user again for a zip code that we did not put in, but doesn't require an input for unit.



Now the Account Management page correctly displays my address. Oh wait! I mixed up a couple of numbers. It's supposed to be 2461, not 2641. It's a good thing we can update this information.



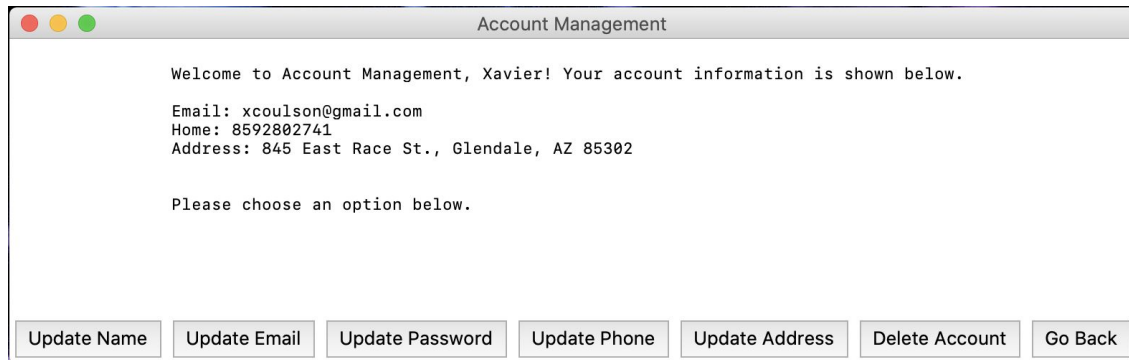
There we go, I was able to fix the address. All that's left to show in this section is the "Delete Account" feature and "Go Back." So let's delete this account after all that hard work!



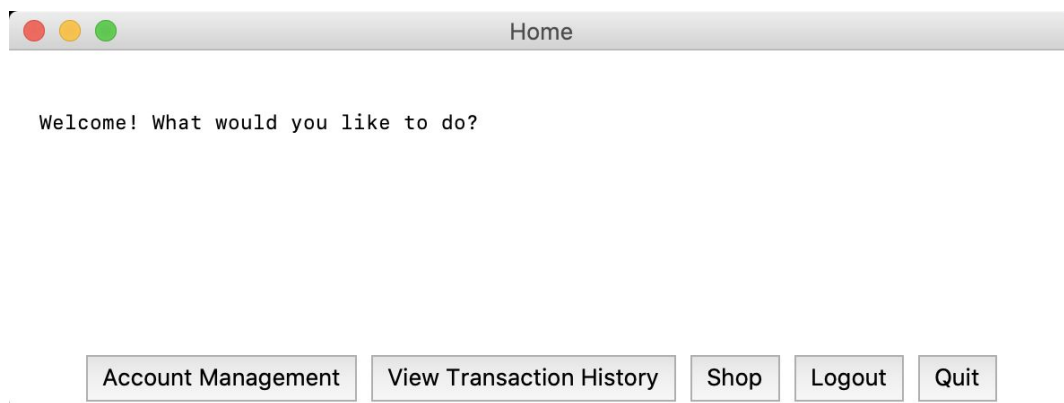
When we delete the account, the program returns us to the main menu. When we try to login with the now deleted account, we are unsuccessful in our attempt. Luckily, I've stolen someone's identity and will use their credentials to log in and continue this demonstration. Say hello to our friend Xavier from Glendale, Arizona!

Note: We do not condone identity theft!

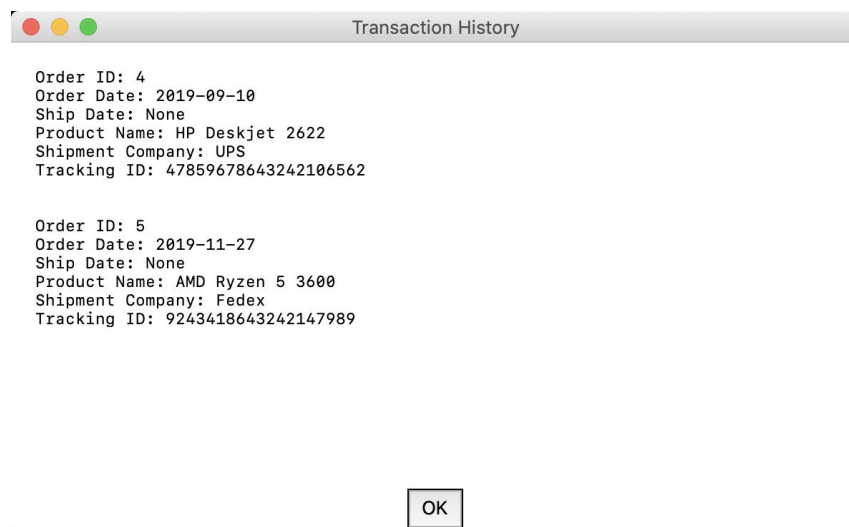




Xavier here has already bought some things from our site, and why wouldn't he? We offer free shipping! Take that Amazon. Everyone gets free shipping. Are we losing money? Yes. But we are stealing customers! Now we finally use the "Go Back" button.

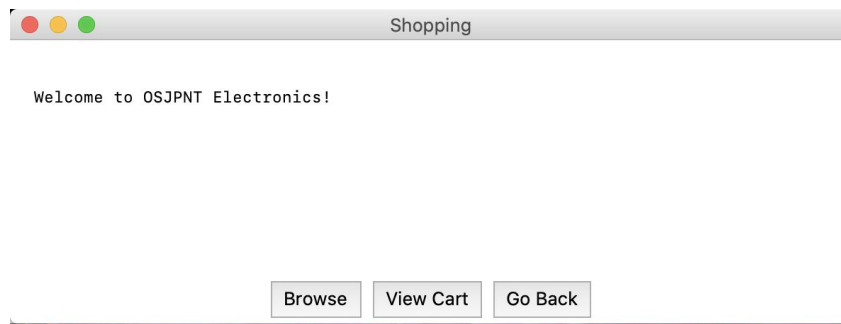


And now we are back to the menu! Let's take a peak at what Xavier has bought by clicking the "View Transaction History."

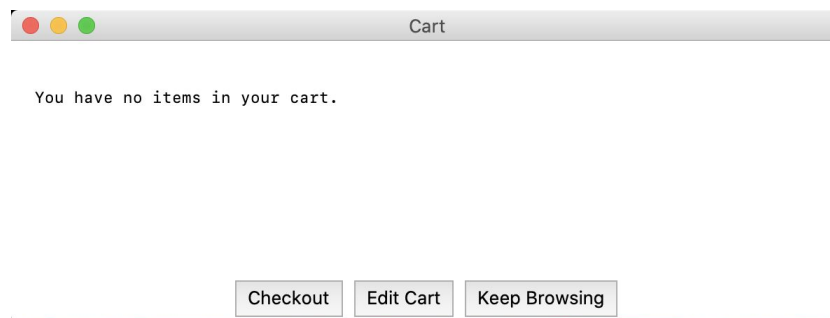


Looks like Xavier here has bought a printer and a processor. Unfortunately it looks like Xavier's stuff hasn't been shipped yet, but I guess that's what happens when you shop somewhere with free shipping. On the plus

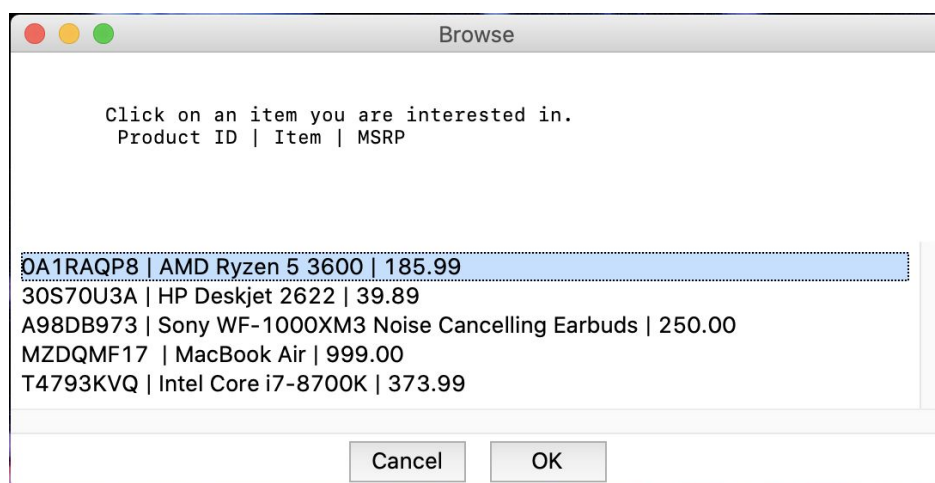
side, when it does ship, we have provided shipping information like who the carrier is and the tracking ID that Xavier can use on the carrier's webpage to see updates on his package. Let's do some shopping!



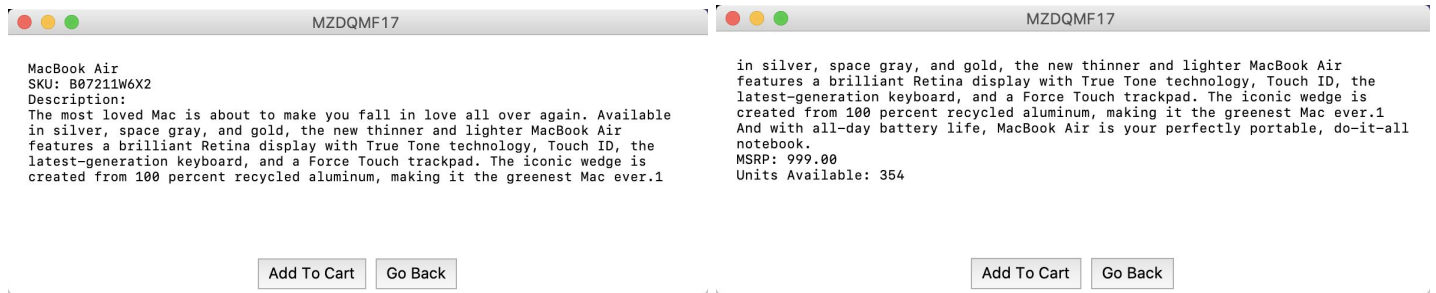
When we click the shopping tab on the menu, we get the box shown above. We have a couple of options when it comes to shopping. We can "Browse" the catalog or we can check out the cart. Let's see if Xavier left anything in his cart.



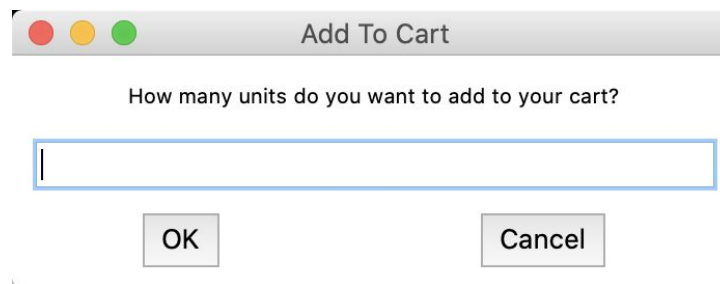
Doesn't look like he did. But that's only because the program is designed to empty the user's cart when they log out and/or close the program. So let's do some browsing.



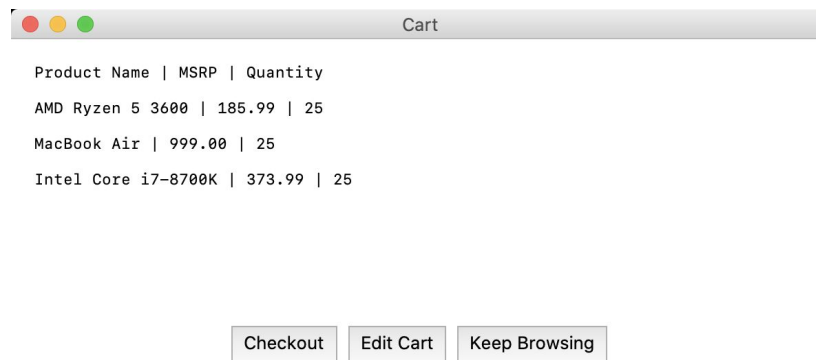
Looks like we have a few options. The first item is that processor that we saw in Xavier's transaction history, and the second item is the printer that Xavier bought in a separate order. The rest of the items are expensive headphones, a Macbook Air, and another processor. Let's say hypothetically we want to buy Macbooks for a class of 25 students, along with two different processors for each student to choose from. So generous!



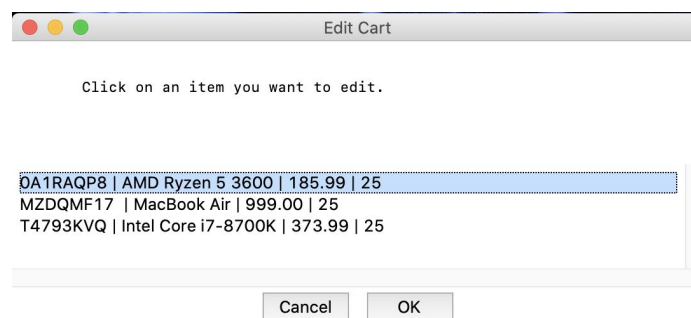
Now when we click on an item, the program brings us to a page that tells us a little more about the product. We can see the name of the product, the SKU, a description, its MSRP, and how many units are in stock. Now we proceed to add 25 MacBooks to the cart, along with the processors.



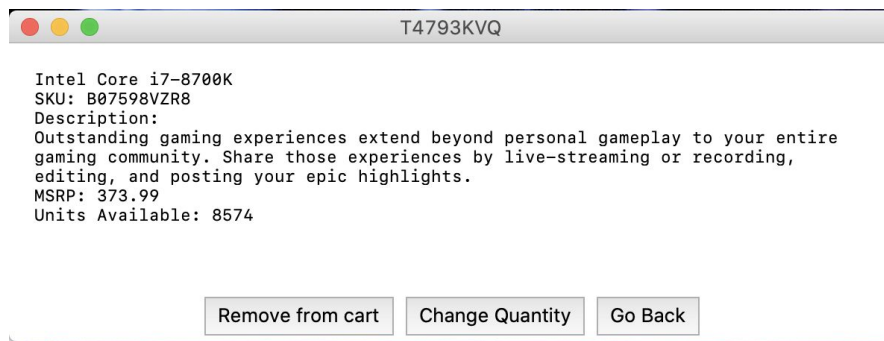
Whenever we add something to the cart, the program needs to know how many the user wants to buy. In this demo, we add 25 Macbooks. Then we add the processors and go straight to viewing the cart.



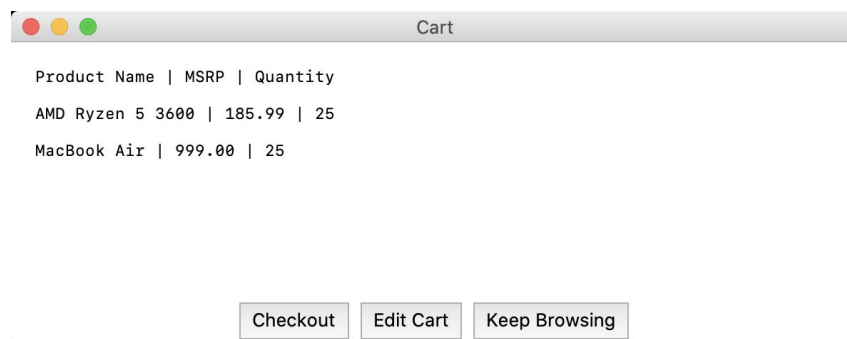
The cart shows our prospective order. Looking at it now, it seems like a lot of money... Let's go ahead and adjust some things. Let's edit our cart.



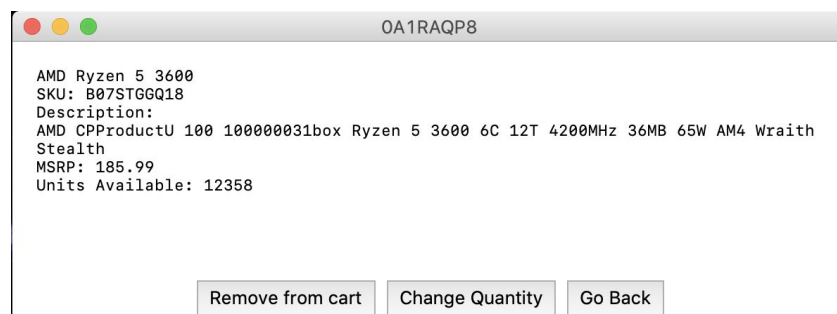
The program now prompts the user to select the item we want to change. Let's get rid of one of the processors.



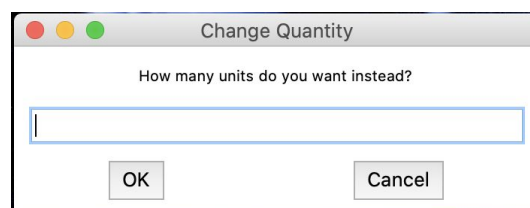
When we select the item, the program will once again give us information about the product. Then we can make a more educated choice. Let's remove the item from the cart.



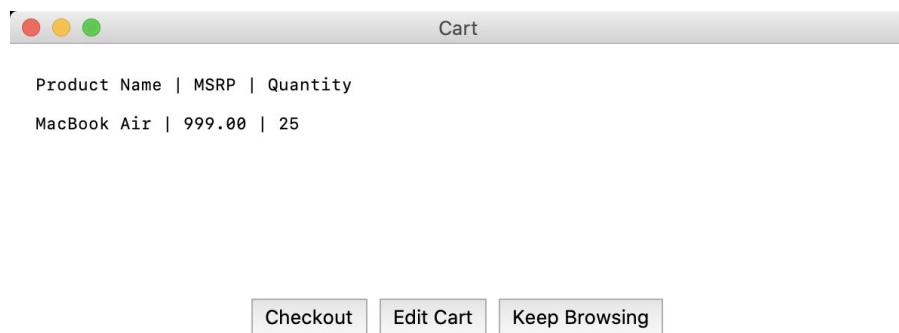
Now that we have clicked remove, one can see that the item was actually removed from the cart, as it is no longer listed in the cart. Let's remove the other processor as well. For demonstration purposes, let's do it via the "Change Quantity" button.



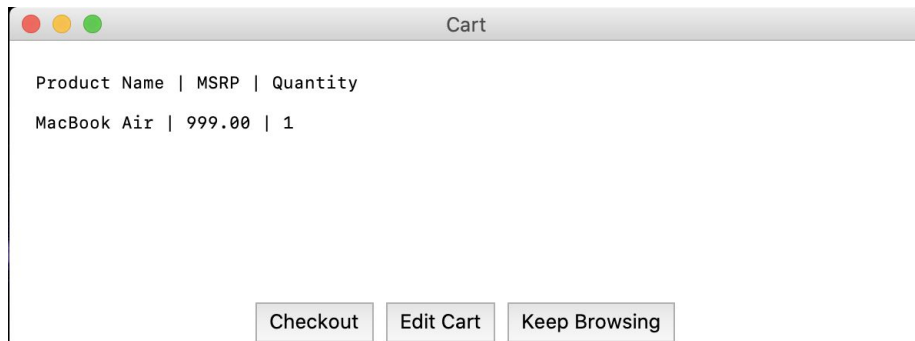
Once again, the program shows us information about the product. This time we select "Change Quantity."



We can set this value to 0 to effectively remove the item from the cart the same way that the “Remove from cart” button did. See the results below.



So it turns out we aren't being generous at all. We are just gonna buy one Macbook Air for Xavier here. With that being said, let's change the quantity to 1.



Now we are down to one Macbook Air!

## Student Contributions

---

Oscar Sanchez → Our project was split into two major sections which were ER Diagram/SQL scripts and Application development. I was responsible for creating the ERD for the database along with the SQL script used to create each table. The initial task of creating entities to support the required functions was a collaborative effort between all members. I also generated a set of sample data used to demo the different features of our application.

Joel Ponder → I helped with ERD and writing functions/queries to drive the application. I set up the RDS database on AWS.

Nicholas Tating → I was primarily responsible for the GUI development, the Application Images section above, and the video demo.