Robbie Decker

Tobias Martin

Avery Hawn

Alex Oleson

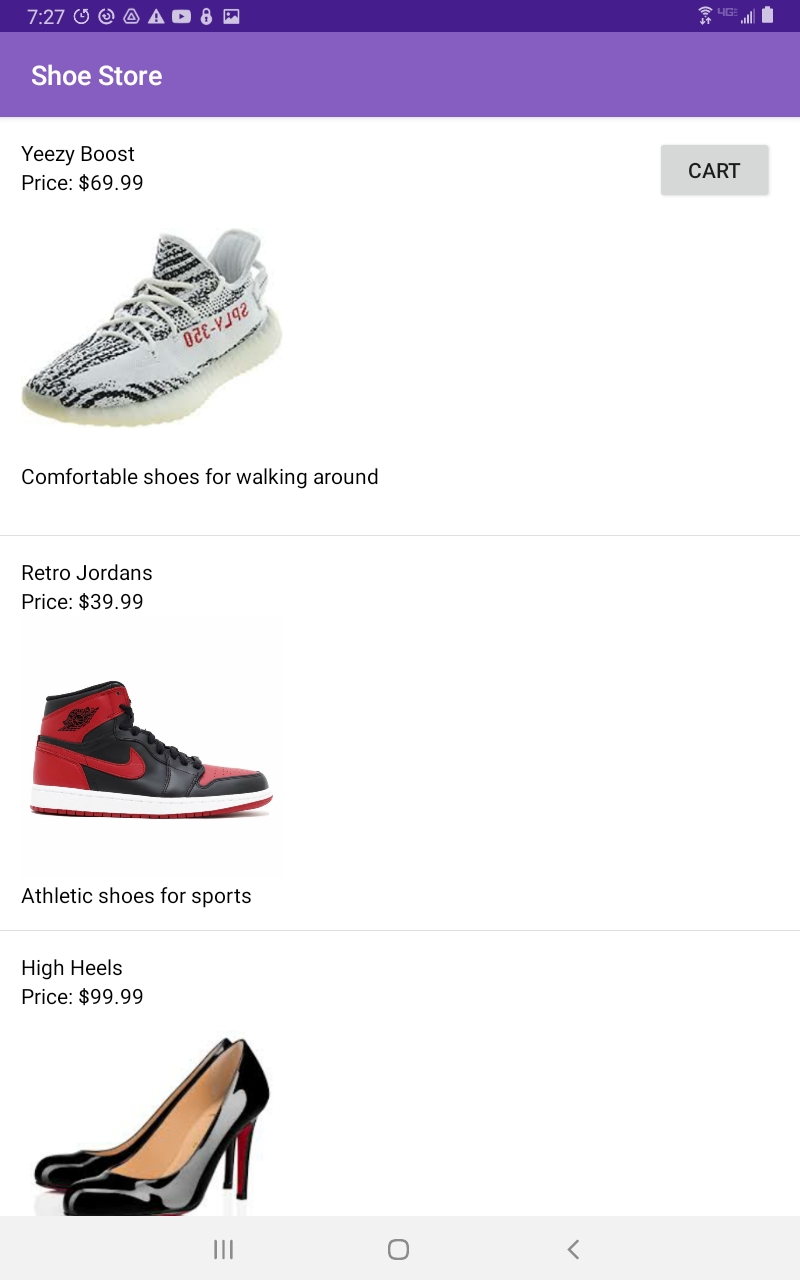
EE408

**EE408 Final Project**

**What Does the App Do?**

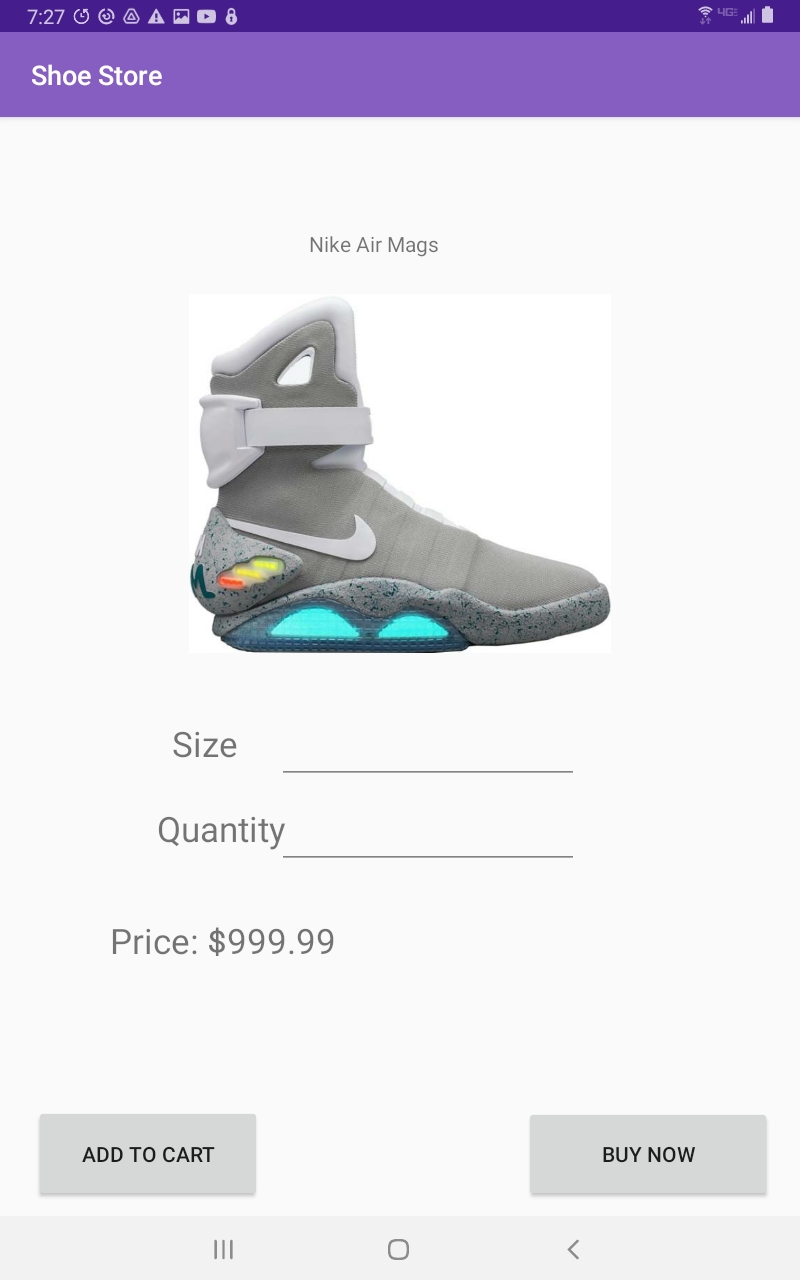
The main purpose of this project was to make a functioning user interface for a person to shop for products at some online store. This includes making a working shopping cart, browsing for items, and paying for the products. Our app was based off an online shoe store. On the landing page you can scroll down to see the different products available. Each product has a price, description, name, and image associated with it.

*Browsing Page or MainActivity.java*



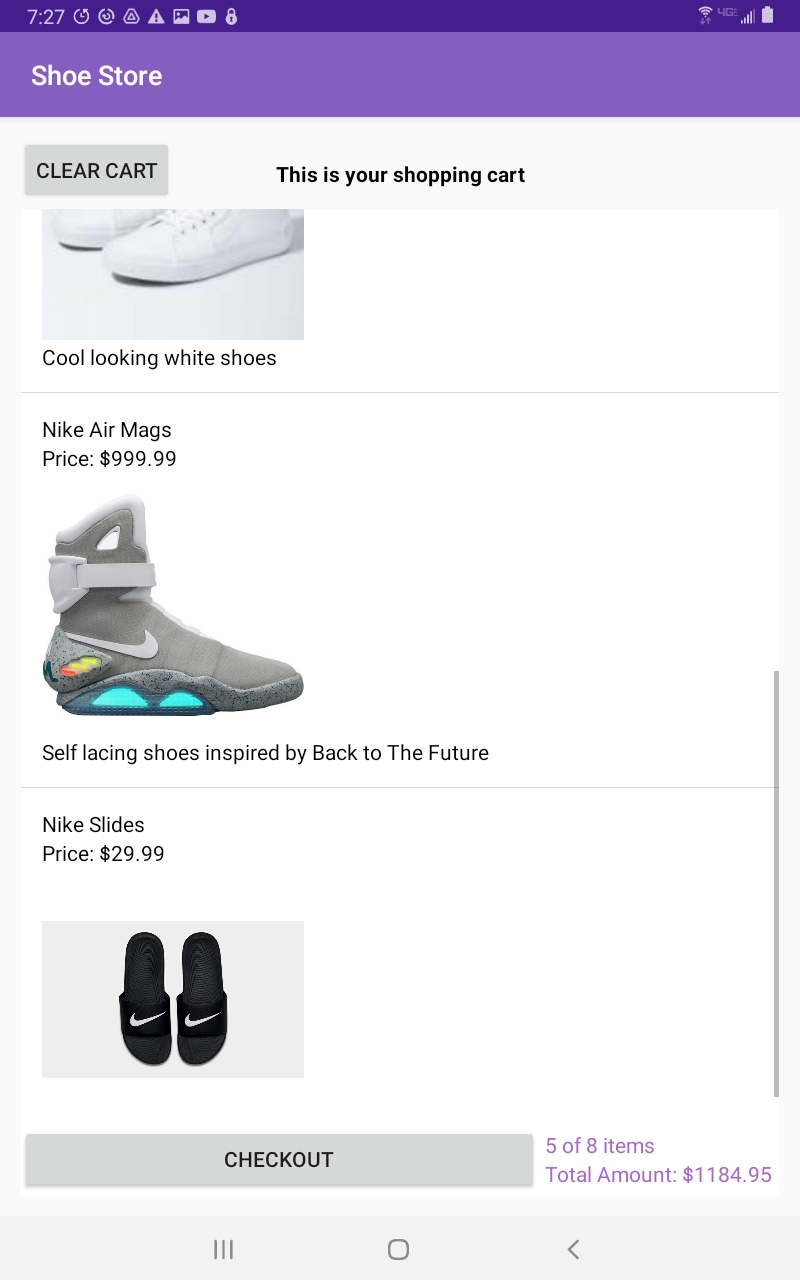
By clicking on the item you are interested in the app will send you to a new screen where you can see the image more clearly. Once here you can input the size of shoe you want, and how many shoes you wish to purchase (as of now the shoe size and quantity feature are not fully developed). The main point of this page is to add the item you are looking at into your cart and continue browsing, or you can buy the item right away. By clicking “add to cart” the item will be placed into your cart, and you will be sent back to the main browsing screen. By clicking “buy now” you will be sent straight to the checkout screen.

*Close Up Page or CloseUp.java*



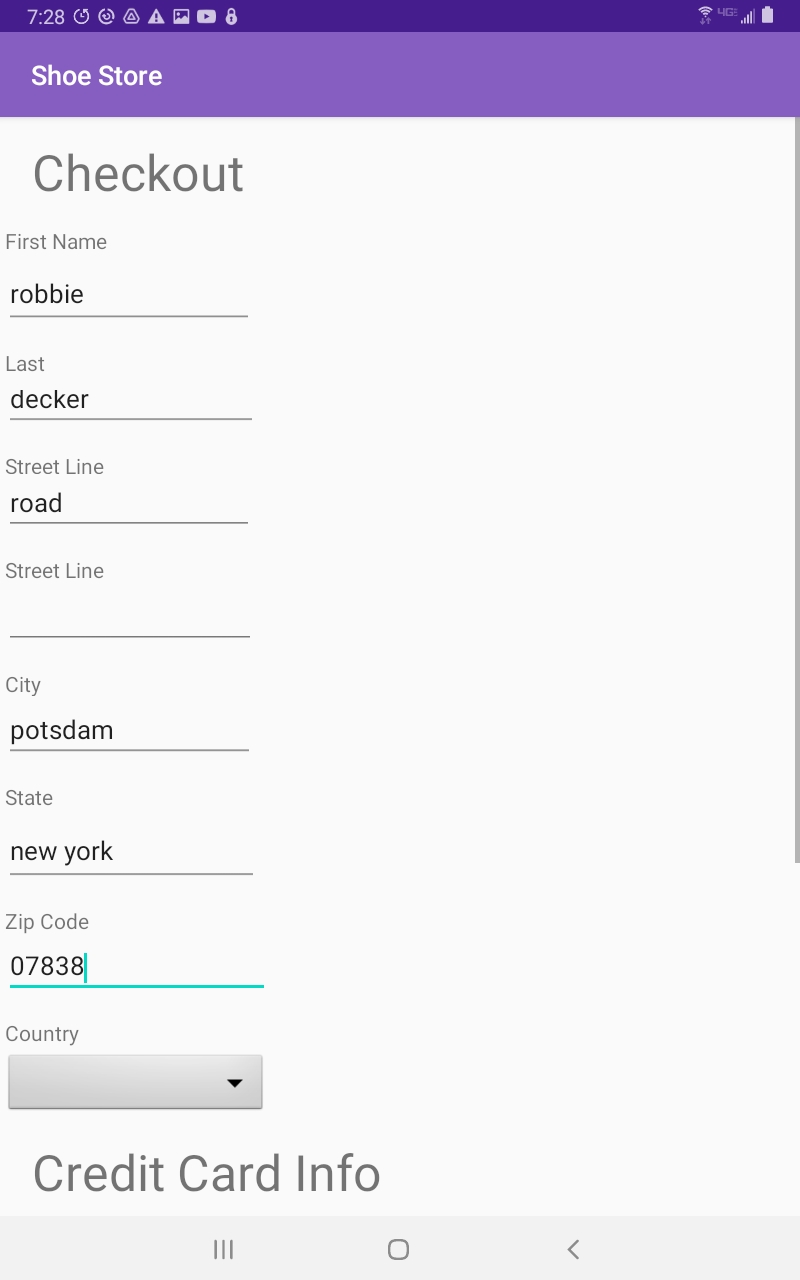
Once all the items you want are in your cart you can click on the cart button on the main browsing page to view your items in your cart. There you can see all the items you selected and the total price of everything. If you wish to edit some item, you can click on the item in your cart and be sent to the same screen that shows the quantity and size of the item. If you wish to buy all the items, you can press checkout from the cart screen.

*Shopping Cart or CartActivity.java*



This will take you to the checkout screen. There you can fill in different information about your name, address, and paying method. At the bottom of the screen is a confirm checkout button which will place your order, but obviously this will not actually buy anything yet.

*Checkout or Checkout.java*



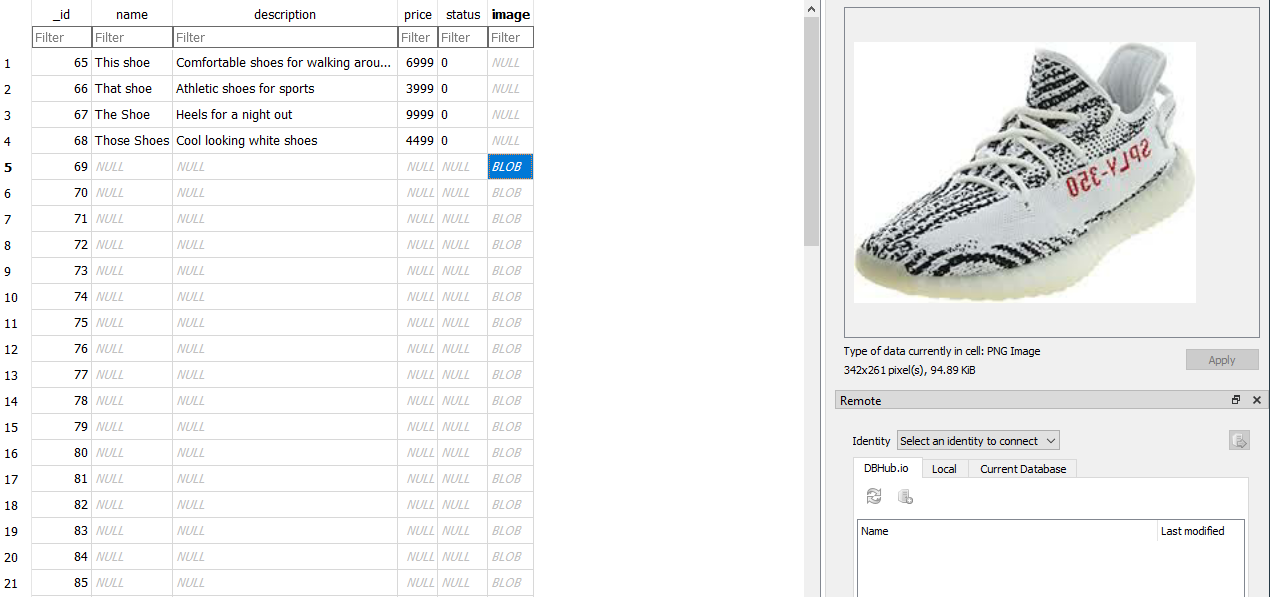
**How Does the App Work?**

There were many different files that had to come together to get this app to work.

* **StoreDatabase.java**
  + The main thing that made this project work was the SQLite database that stored all the products. The main table was called “shop” which had six different columns: id, name, description, price, status, and image. The id was a number to keep track of the items, the name was the name of the shoe, the description was a small sentence that described the shoe, the price was the price tag associated with the shoe, the status was to keep track if the shoe was in your cart or not, and the image was the picture associated with the shoe. There were many helpful methods in this file such as createItem which would insert information into the database, and deleteAllItems which would clear the database.
* **MainActivity.java**
  + This is the main browsing page which creates each of the items that are shown which means that they are added to the database. The images used for the products first need to be bitmaps before being sent to the database. The next important thing this does is show the products in a listview. Getting this data to show the right way was somewhat difficult. A SimpleCursorAdapter was used to show the data, and a cursor was used to go through the database to get the right information. Another important part of the listview was setting a new ViewBinder (MyViewBinder.java) which put the data into the listview the right way. The ViewBinder was needed to get the image to display correctly. The last thing this file dealt with was what to do when someone clicked on an item in the list view. The function onItemClick dealt with this. It would set a new cursor to the right row of the database table of the selected item. The cursor would then be used to get the name, description, price, id, and image. The image would be received as a byte array. A new intent was made and all of the data was added into this intent via the intent.putExtra() method. Doing this would pass the right data to the close up screen.
* **CloseUp.java**
  + The file starts by getting the previous intents data. All of the TextViews and ImageViews are then given the appropriate data from the intent. This allows the right name, picture, and price to show up for the product. This screen also adds items to the shopping cart by changing the status of the item to 1 which helps the Cart decide what has been added and what has not. Not added items have a status of 0 and added items have a status of 1. The “buy now” button still adds the product to the cart but has an intent that changes the screen to the checkout screen.
* **CartActivity.java**
  + This uses the same system for the ListView as the MainActivity file does. It also uses some functions in StoreDatabase to show how many items out of the entire store you have in the cart and the total cost of all of the items. To get the number of items in the cart we use queryNumEntries where status is a 1. To get the total number of items in the store a cursor is used to go through the rows and get the number of rows used in the database. To get the total cost in the cart all of the prices of the items with a 1 status are added together. Like the MainActivity screen, this cart also allows the user to tap on an item and go back to the close up view of the product. This also has a button that has the intent to send the user to the checkout screen.
* **Checkout.java**
  + This mainly acts just to show the xml for this page. It does also make a new instance of the object Person to store the information about the person and their card information.

**Problems faced**

This project was a big struggle for a couple of key reasons. Getting the cart system to work for a while was very difficult. The main problem was working with SQLite databases. Everyone in the group was new to any kind of databases, so it was hard to get the data to store in it, and then access it the right way. The biggest problem faced was working with the images in the database. At first this project created the items in the database in the StoreDatabase.java file which did not work when it came to putting images in the database because it could not access the resources in the project. To work around this we tried adding the images to the database from the MainActivity.java file, but this resulted in the pictures not being added to the right rows because I think we used two different instances of ContentValues to import data into the rows. This meant that new rows were being created for the images instead of adding the images to the already existing rows. The null images can be seen in the image below.

*Null Images in the Database using DB Browser for SQLite*

The main way I found out about this was to go to the device manager in Android Studios and export the shop database used. Then I downloaded DB Browser for SQLite which showed the table in an easier way. This program was used to get the screenshot below. The work around for the problem was to just create the entire items name, description, status, price, and image all in the MainActivity.java file. On top of this, converting the images back and forth between bitmaps and byte arrays were also trivial at times.

**Improvements to be Made**

We think this project has made great strides since it was shown during the presentations class, but we also think a lot of the UI needs to be polished. Some added colors would be nice. In addition, we would like to mainly get the quantity input field to store the number of items being bought. The shoe size functionality also needs work. In a more fleshed out app, the checkout behavior would also have to be expanded on to make sure that everything is input the right way. Overall, this project did come out much better than when we showed it but could still be improved upon.