Name: Bryan Heiser

Project Description: Winter Sports Equipment Store. Store for users to shop for their winter gear. Viewing overall items, viewing individual items, add items to cart, and purchase items.

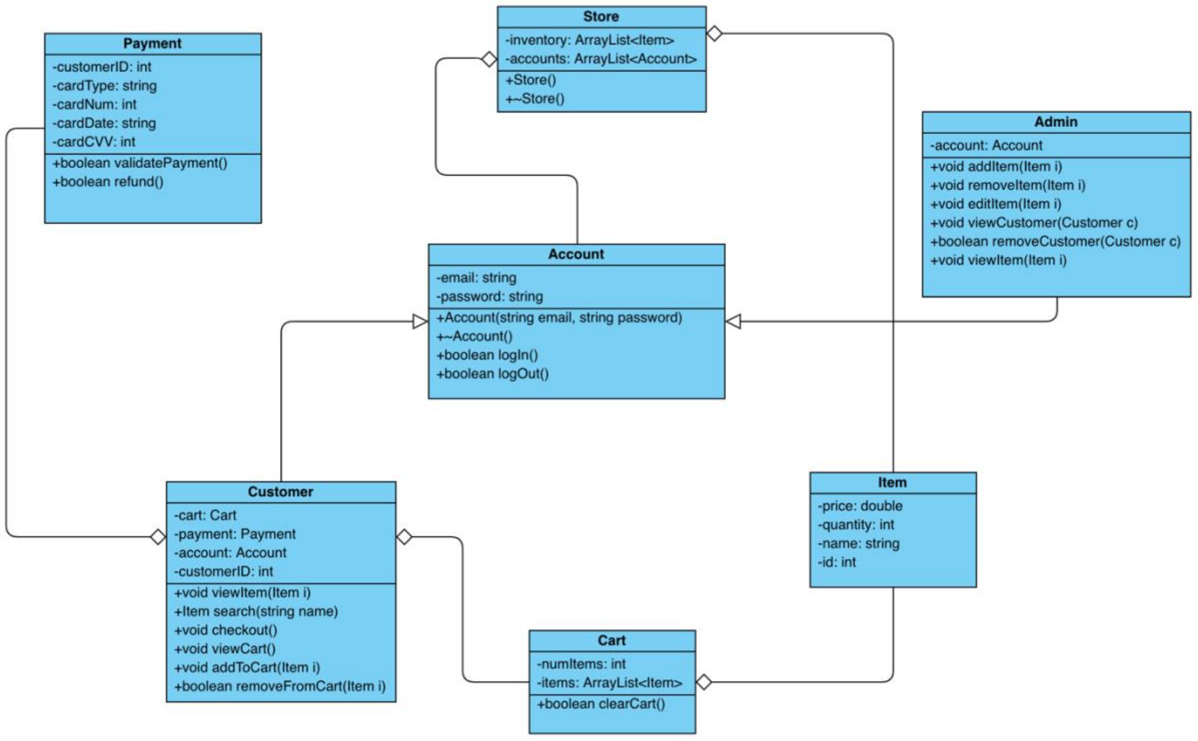
Implemented:

User Req. ID	*User Requirement*	*Description*
UR-03	Customer can view an item	Customers must have either searched for an item and clicked to on it.
UR-04	Customer can view their cart	Customer must be logged in to their account.

Not-implemented:

User Req. ID	*User Requirement*	*Description*
UR-01	Customer can add an item to their cart	Customer must be logged in to their account and currently viewing the item they would like to add to their cart.
UR-02	Customer can remove an item from their cart	Customer must be logged in to their account, be viewing their cart, and their cart must contain one or more items.
UR-05	Customer/Admin can search for an item	There must be a match between product names and search input for results to be displayed.
UR-06	Customer can create an account	Customer must have a valid email address and meet the password requirements.
UR-07	Customer can delete their account	Customer must have an already have an account and must be logged in to do this task.
UR-08	Customer can checkout	Customer must be logged in to their account, have at least one item in their cart, and enter a valid form of payment.
UR-09	Customer/Admin can log in	Both users must have an account.
UR-10	Admin can view a Customer's account	Admin must be logged in and the customer must have an account.
UR-11	Admin can remove a Customer's account	Admin must be logged in and the customer must have an account.
UR-12	Admin can add items to inventory	Admin must be logged in.
UR-13	Admin can remove items from inventory	Admin must be logged in.
UR-14	Admin can adjust price of an item	Admin must be logged in and the item must exist in inventory.

Class Diagram:



This was my class diagram that I anticipated on implementing. I set up my database and classes to represent these relationships. I did end up dropping Account and just gave users their own email and password variables.

Design Pattern: I would have liked to use State for this project. This would allow for individual pages specific to the user when they log in or sign up. This would have been done by creating html pages that were altered by the state of the user currently interacting with the program.

What I have learned/final thoughts:
This project had a huge learning curve due to my lack of experience with web development and especially frameworks. My initial strategy was to first complete all the frontend development, anticipating this to be the most difficult aspect for myself. But after fully implementing the frontend side, I did not plan for nearly enough time to complete full processes I intended. I was able to set up the database for the backend, create tables, and input information. I was able to create my classes, their attributes and how they work together. However, the middleware of this project was the most challenging and complex part. I used Django for my framework, which provided far more starter code than I knew what to do with or how to use properly. There was a lot of help online, yet I still found myself in circles attempting to implement features and functionality in this framework for my specific project needs. I am very disappointed with how much I was able to implement given the time period for this project and regret my assumptions

about this project I made throughout its development. For the future would spend much less time on the frontend and worry almost entirely about the middleware and backend that constitute the functionality of the program.