**Project Summary – Shopping Cart App**

In this project, Python was used to create a shopping cart application. The application is currently designed for a sample healthy foods store, with 4 products with various prices and inventories, based on the prepopulated data in the project database. PyCharm was used as the development environment of choice, and the project GUI was generated with the assistance of Python’s TKinter. Overall, the application takes several text and button click inputs from the user to complete several shopping cart actions, including browsing items, sorting data, searching data, displaying information, adding and removing items from cart, and ultimately, checking out.

The shopping cart app code is broken into three main sections: business, database, and presentation. In the business file, all of the classes that operate on the data from the database, whether it’s to calculate, modify, or display, are declared. In the database file, all of the sqlite commands and connections to databases and tables are made. Finally, in the presentation module, all of the functionalities provides by the two other modules are incorporated by a main function, and all display functionality is controlled in this file as well. Separating the project into modules and classes made it easier to compartmentalize the code, debug/troubleshoot, and handle all of the data and operations.

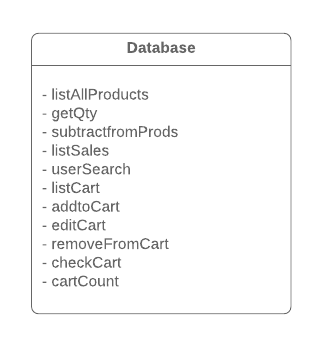
When the code is first executed, the welcome page can be viewed; here, all of the products in the store can be viewed, sorted, and searched based on the user’s preference. Welcome messages are also displayed, and there is a cart button that updates every second to display the current amount of items in the cart waiting to be ordered. Users have the flexibility to add an item to the cart singularly, or in a desired quantity. If there are not enough items in stock, an appropriate message will be displayed and none of those items will be added to the cart. Moreover, any items that are sold out will have an inactive Add to Cart button, in order to avoid the creation of an item order backlog. Once in the cart, items can be either added or subtracted in any quantity defined by the user. In addition, the subtotal, tax, and total can also be viewed in the cart at all times. When a customer checks out, not only is there an appropriate message that displayed, but also, the items that were ordered are removed from the product inventory database accordingly, and the cart is cleared.

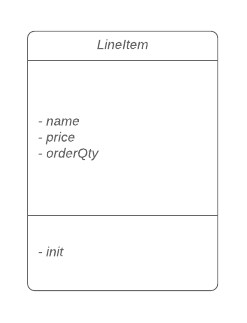
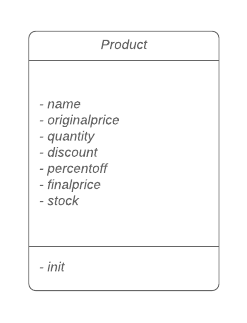
During my time writing the code, I did encounter a few moments where I was stuck trying to get my code to perform what I was imagining. One of the first lessons I overcame was learning to not mix grid and pack formats for my display, and determining which formatting method was most useful for the way I imagined my app looking; originally, I thought grid would be the best method, however, pack proved to be the method that produced the best results based upon my GUI conceptual design. Other small miscues included database problems, such as attempting to write to my database when it was opened and unsaved in sqlite browser, or not fully saving preloaded data into the browser. After tinkering with the program a little, I overcame these issues with no problem. Later on, I solved more complex issues, including using lambda (anonymous function) to call a function with parameters—which was useful for button commands that take functions with no parameters—and creating a refresher function to check the status of a value of the database (or a calculated value based on database conditions) and update my GUI accordingly at regular time intervals (which reminded me of something I had done for a website project in Javascript for my Intro to Internet Computing class). I enjoyed coding the logical statements that enabled the program’s main functionalities and display processes; it was satisfying to gradually see my code develop into what it needed to be to display the desired output.

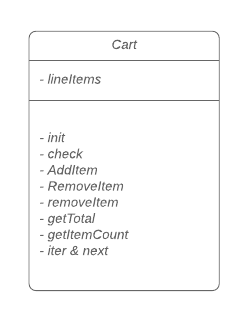
All in all, the program performs the basic functions that one might expect from a typical shopping cart. While the project runs properly and meets the minimum design requirements, there are some considerations for further development in the future. To begin, a web-based version of this app would be interesting for online shopping support. In regards to making the app itself more robust, more research and development to improve the display might be beneficial, in order to make the GUI more aesthetically pleasing and intuitive; default values for drop down menus and photos to accompany products would very likely improve the user’s experience. Functionality-wise, implementing a fuzzy search instead of a strict search would be a huge improvement, since currently user searches will only return results of exact, non-case sensitive matches. Incorporating the discount/sale data in the program is also an idea for future development, possibly to include a “total savings” metric for display at the cart or at checkout.

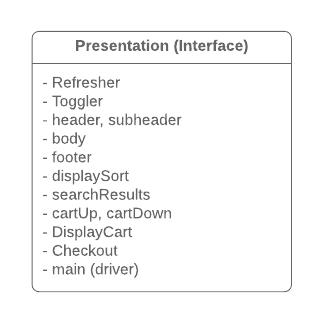
**UML Diagrams**

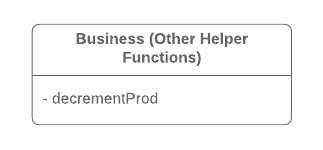
* **Database**
* **Business**
* **Presentation**







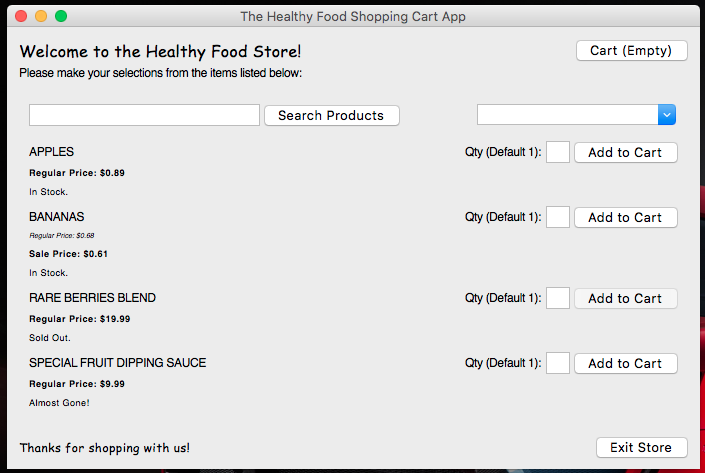




**Application Flow**

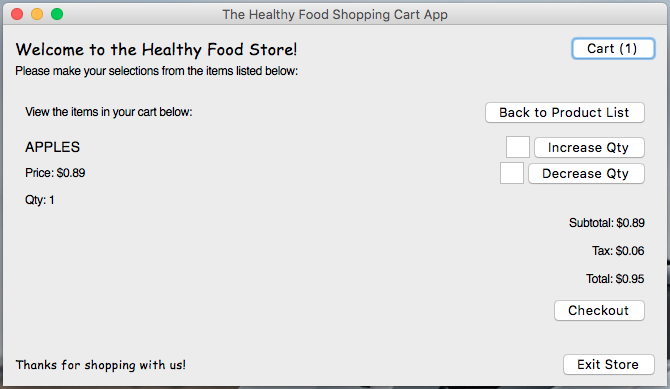
Home Screen

* View all items in store
* View number of items in cart
* Sort and search items
* Exit Store (option)
* **Next Steps**
  + **Add Items to Cart**
  + **Click Cart Button**



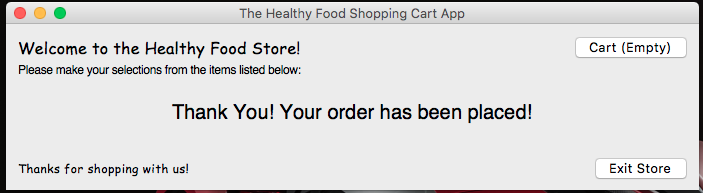
Cart Screen

* View all items in cart
  + Price
  + Quantity
* View cost breakdown/totals
* Add/remove items from cart
* Return back to product list (option)
* Exit store (option)
* **Next Step: Checkout (Click checkout button)**

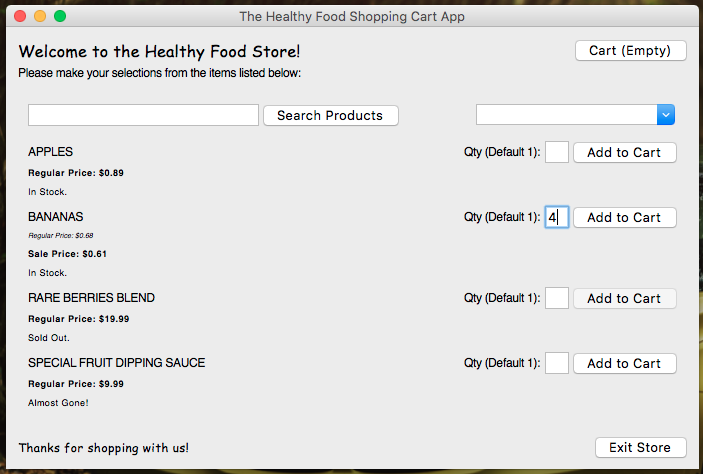


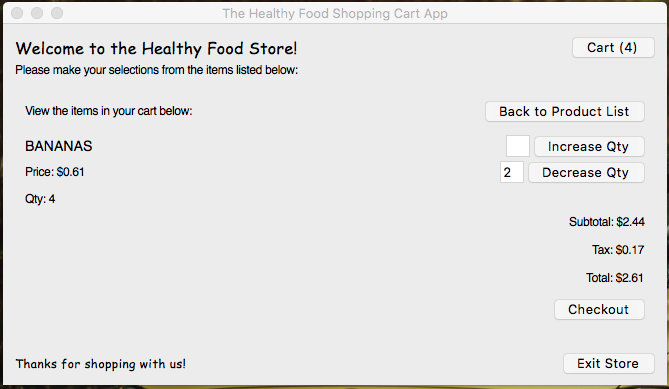
Checkout

* Get feedback regarding your checkout
* **Next Step: Exit program**

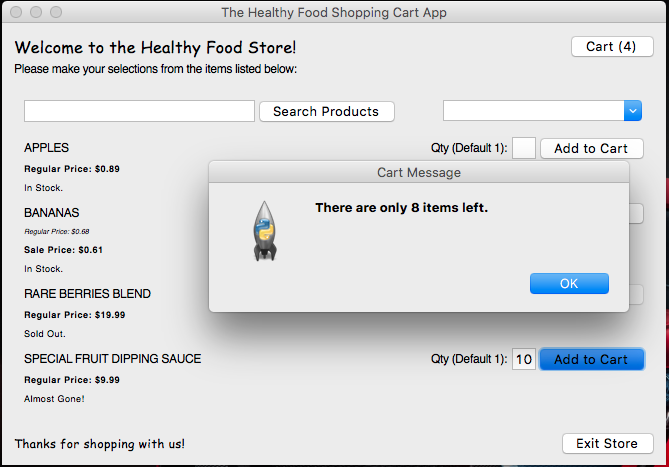


**Sample Use Cases**

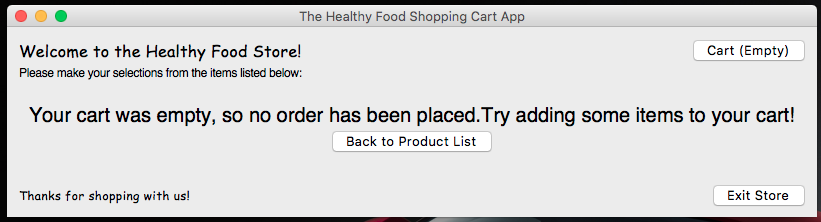
* Add multiple items to cart from home screen
* Enter value and then click “Add to Cart”
* Remove multiple items to cart from cart screen
* Enter value and then click “Decrease Qty”
* Conversely, enter value and click “Increase Qty” to add more items



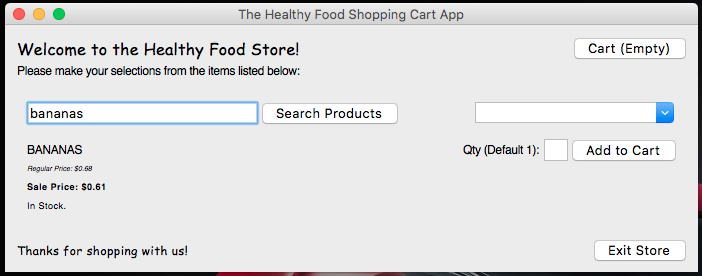
* Accident: Adding more items than in stock



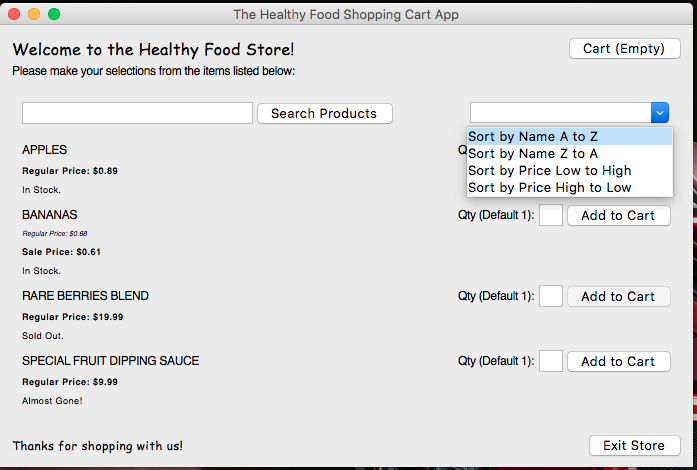
* Accident: Checking out with an empty cart



* Search for items to add to cart at home screen
* Enter value in search box, and then click “Search Products”



* Sort items to add to cart at home screen
* Select sort type from dropdown



* Exit program
* Use “Exit Store Button” or window close button on any page

../../../../../../Screen%20Shot%202018-12-02%20at%2011.03.26%20PM.p or ../../../../../../Screen%20Shot%202018-12-02%20at%2011.03.33%20PM.p

**Links**

GitHub: <https://github.com/djricky5/HFSShoppingCart>

IBM Cloud

* <https://git.ng.bluemix.net/nelsonr2015/Nelsonr2015HFSShoppingCart>
* <https://console.bluemix.net/devops/toolchains/09030777-4d30-4a72-84a8-a6f63c3dd043?env_id=ibm%3Ayp%3Aus-south>