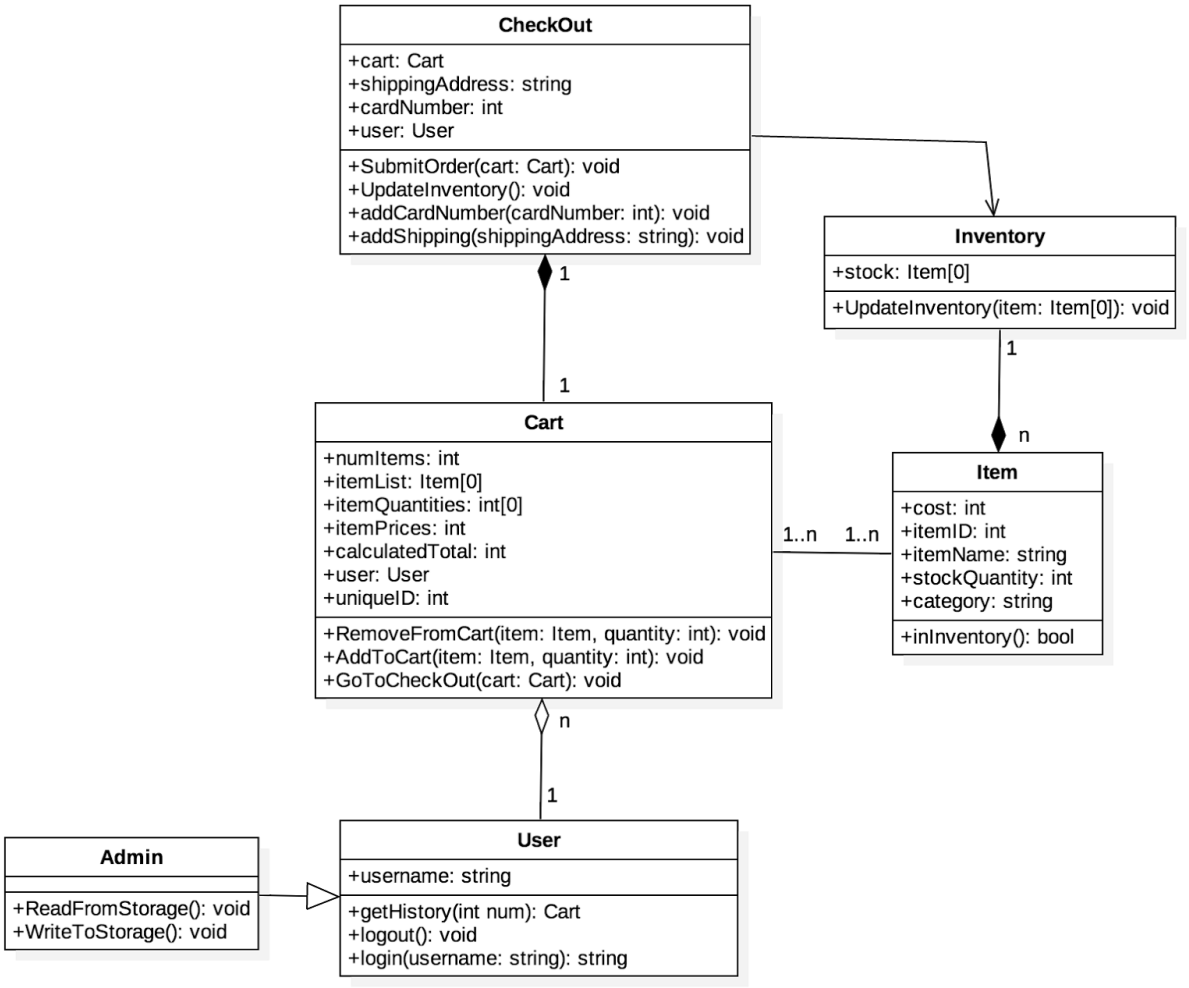
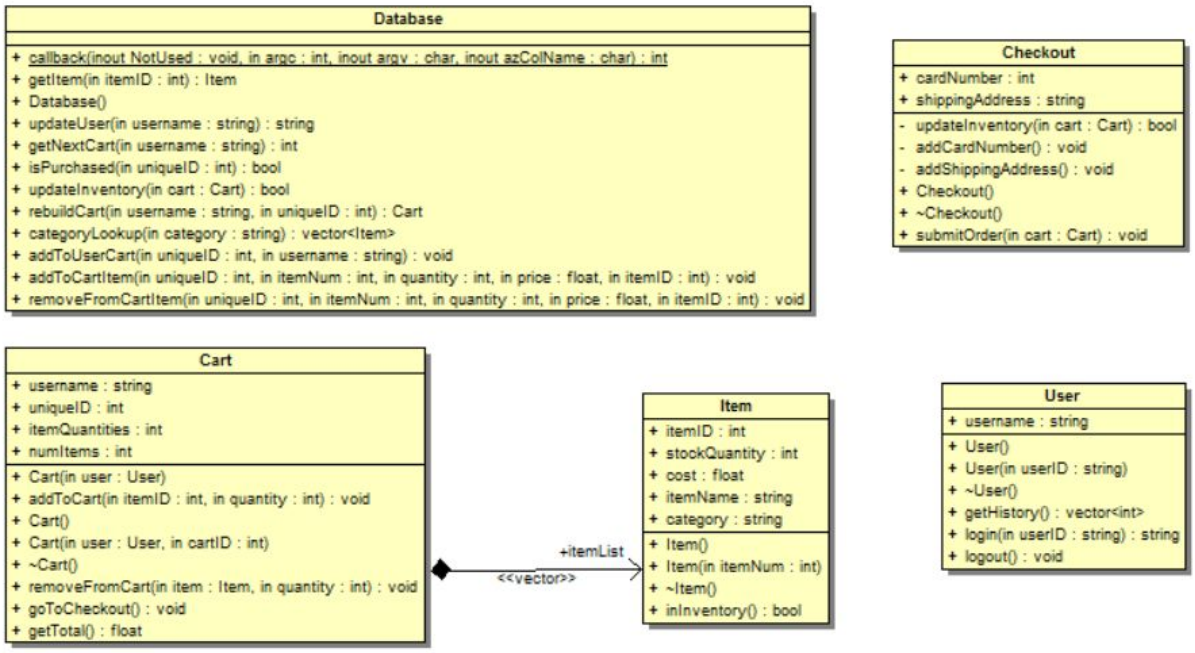
GitHub Repo: https://github.com/tmwinstead/CSE4233-Quiz03

Class Diagram from Quiz 02

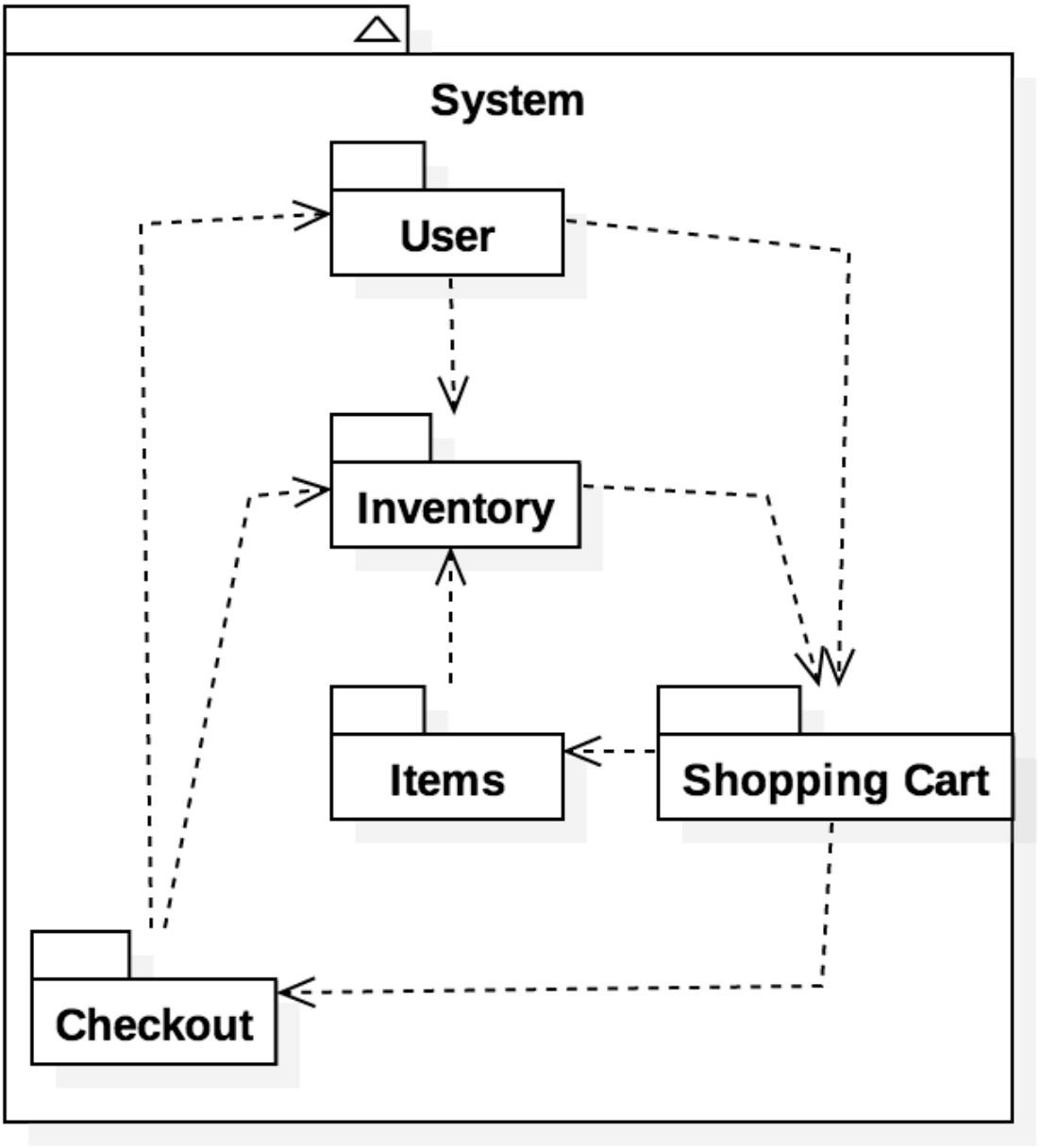


Class Diagram after Implementation

We were unable to generate a class diagram with StarUML, which generated garbage, but we were able to generate a diagram with another tool. Here is one from a tool called BOUML:

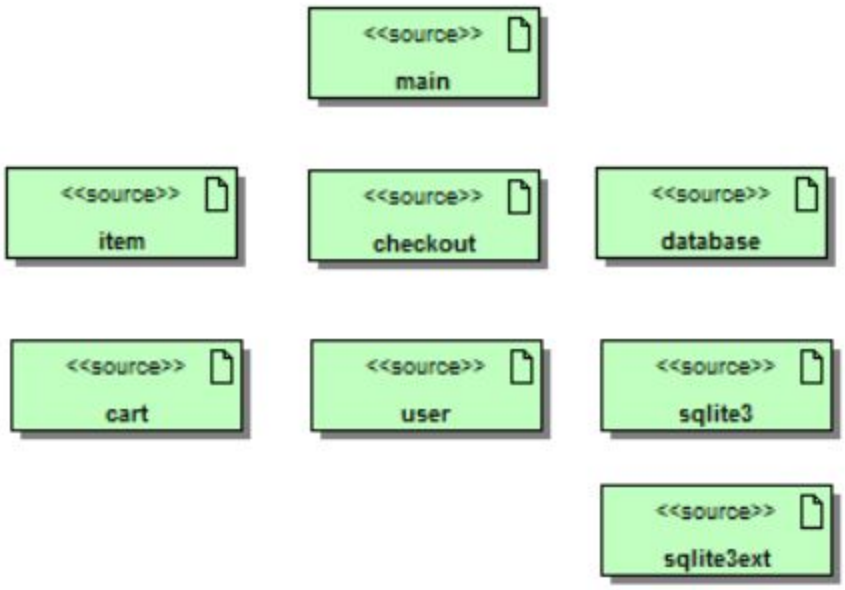


Package Diagram from Quiz 02



Package Diagram after Implementation

As before with the class diagram, StarUML still generated garbage; therefore, we were unable to generate a package diagram with it. We were able to generate one with BOUML:



Conclusions

We had a very difficult time implementing the actual code for the prototype. At the time of this writing, we can still not execute the prototype successfully. One of the biggest challenges was not the classes or logic, surprisingly. The hardest part we have encountered trying to get our system to work is the databases. Our prototype is written in C++, however, the SQLite3 files are in C, which we believe is part of the problem. While the two are compatible, making them work together is proving more than challenging. We have spent countless hours (but if we had to count, 25+ hours, each) writing, editing, and compiling, only to have the results vary from not getting past the initial execution to getting several functions into execution before failure. The inconsistency makes it extremely hard to debug, resulting in lots of confusion about what is even broken. As far as creating the database goes, it seemed straightforward at first; however, we estimate that the database is only successfully compiled 4 out of 5 compile attempts - for no particular reason.

On another note, the class diagram from quiz 2 helped guide us initially as a basic blueprint, but we implemented many changes throughout the development process. In our planning stages during quiz 2, we focused on how objects could potentially interact with other objects in order to create many connections. Many of the initial variables remained and are in the final version. Upon implementation though, we realized that all of the various classes were not necessary and were each somewhat isolated, resulting in a lack of connections. We started out with the same variable names from our initial diagram, and then we changed those variables and each class' methods as we coded all other classes. Most changes were the result of realizing each class' role in the overall system and then changing the methods to work for our implementation.

Because of these implementation changes, the diagrams look very different. This is due to the fact that we saw it was unnecessary to include an entire object in most class constructors, as all we needed was one property of the required object.

Using two different machines, we got two completely different databases resulting from executing the same code, with the same compiler. Both of these machines were on macOS. With one user, terminal would segmentation fault more often. On the other machine, the database ended up being entirely different. We confirmed that the two buildDB.cpp files were the exact same with an MD5 hash. However, the databases still ended up being different.

We would not recommend applying the StarUML RE plugins, as they returned mostly blank forms.