Project Title: Dashboard for Inventory Management (DIM)

Description:

I will be creating a GUI using Django for Python that displays information about a database of inventory item and customers for a business. My example business will be the company I work for, Sutton Garten. They are a welding and dry ice company. The parts of the inventory I will focus on first will be the bottles (of gas) and the tubs (of ice). These are the two main items that get filled, shipped, sold, returned, rented, emptied, etc. I will use classes to create the database structure as well as create objects in my python code. I will create a way to add tubs and bottles to the database, while I make sure they are unique and not duplicates. I will create a way to dock tubs and bottles full, empty, shipped, returned, out of service, etc. I will use the various pieces of information on the tubs to display graphs and charts to the user about the company’s assets as a whole. I will start small and add features as fast and often as I can before the deadline, and I will submit whatever stable version I will have created by then.

User: Sutton Garten Employee(s)

Problem: Poor Inventory Overview Abilities

Technologies: Python, Django, SQLite, HTML, CSS, Bootstrap

Case Analysis:

Sales Rep

--Adding Customers

--Updating Customer Files

--Creating Orders

--Processing Payments

Warehouse Labor

--Adding Items

--Filling Items

--Shipping Items

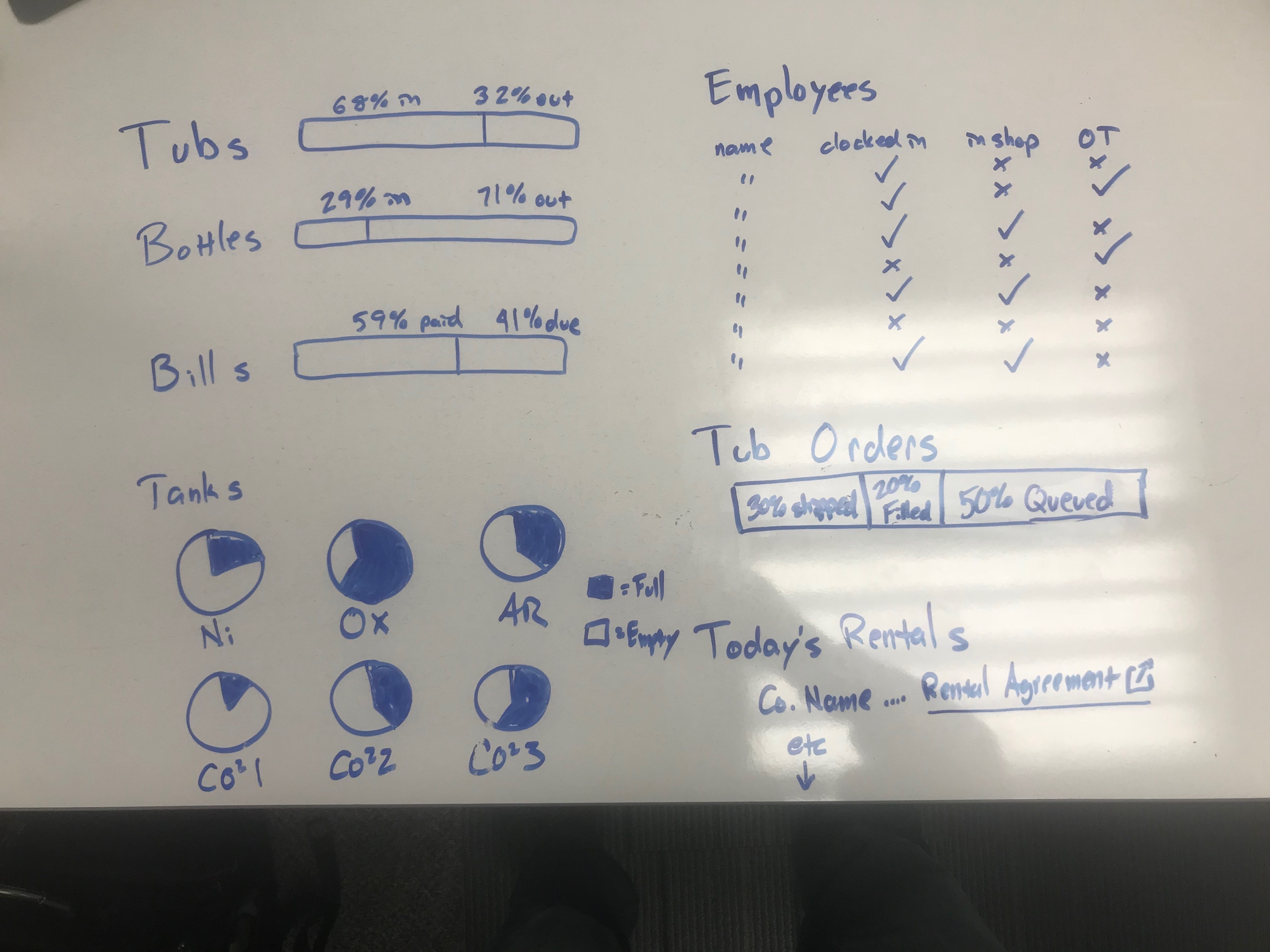
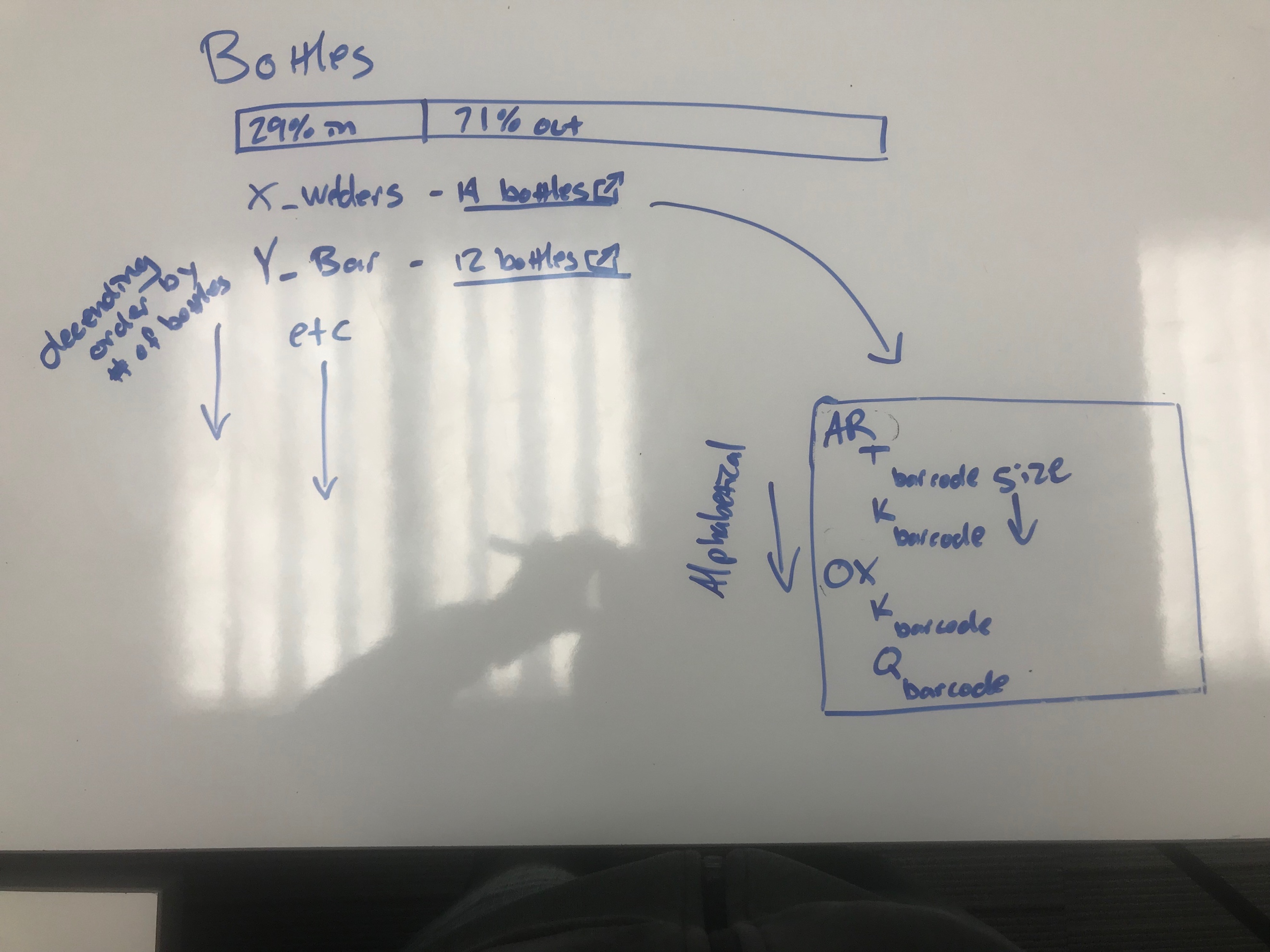
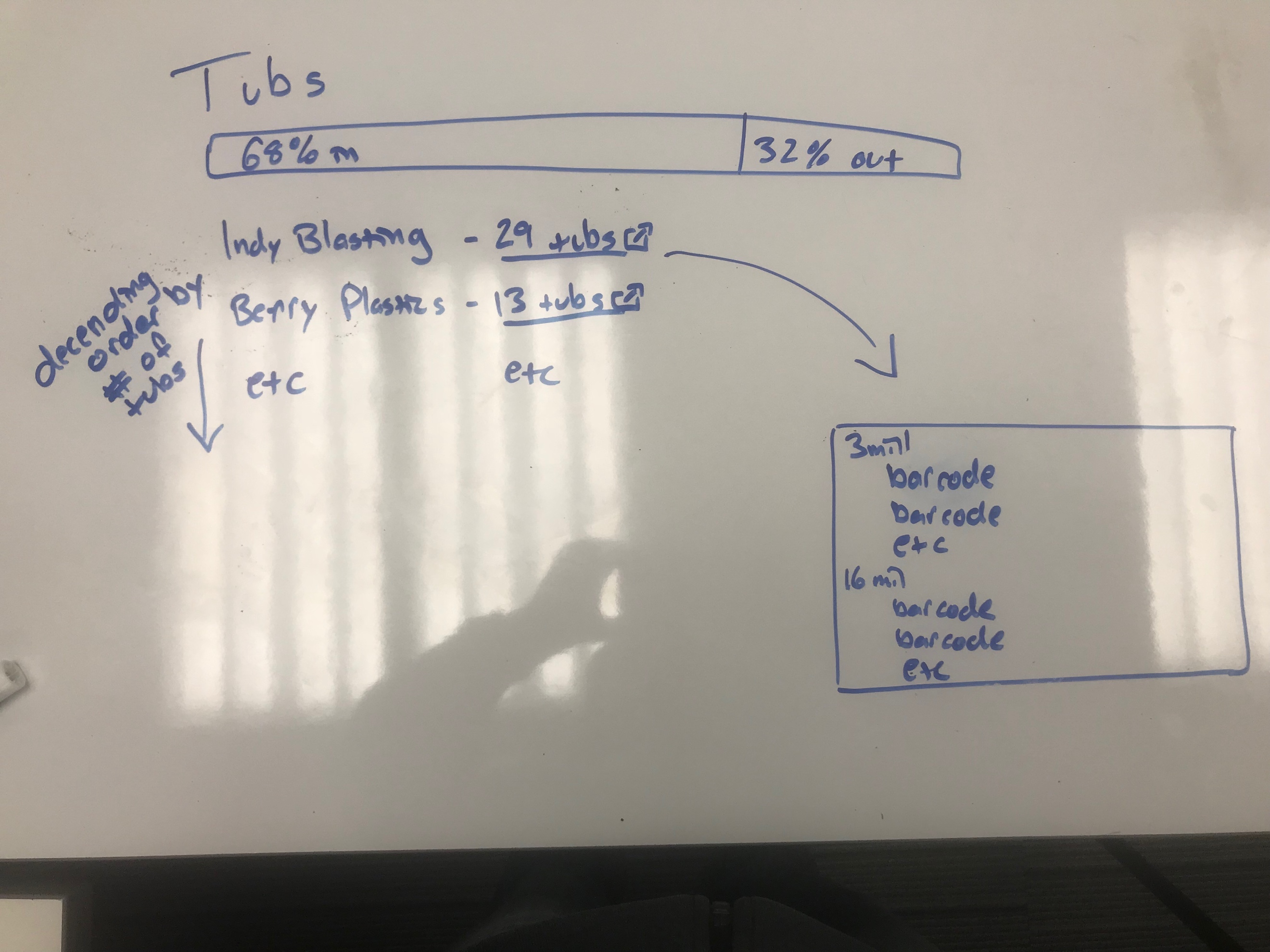
--Receiving Items

Management

--Viewing Dashboard

Data Design: Django’s built in class models for databases

UI Design:

These are some initial views. Like I said, I will be adding as much as I can before it is due.

Algorithm:

There is no way I will be typing out my whole methods in psuedo code. I will try to update this as I go though. For now what I have is:

Create a url path to a main page. Create a view that returns that main page. Create that main page. Pass in values to that main page. Use css/bootstrap to style the main page to display information.

Create a url path to add an item. Create a view that returns the form to add that item. Create the form page. POST the values back to the database and save it. Redirect back to the main page. Repeat this for any add, updating, or deleting of pages.