CompSys 704

Project 1 POS System

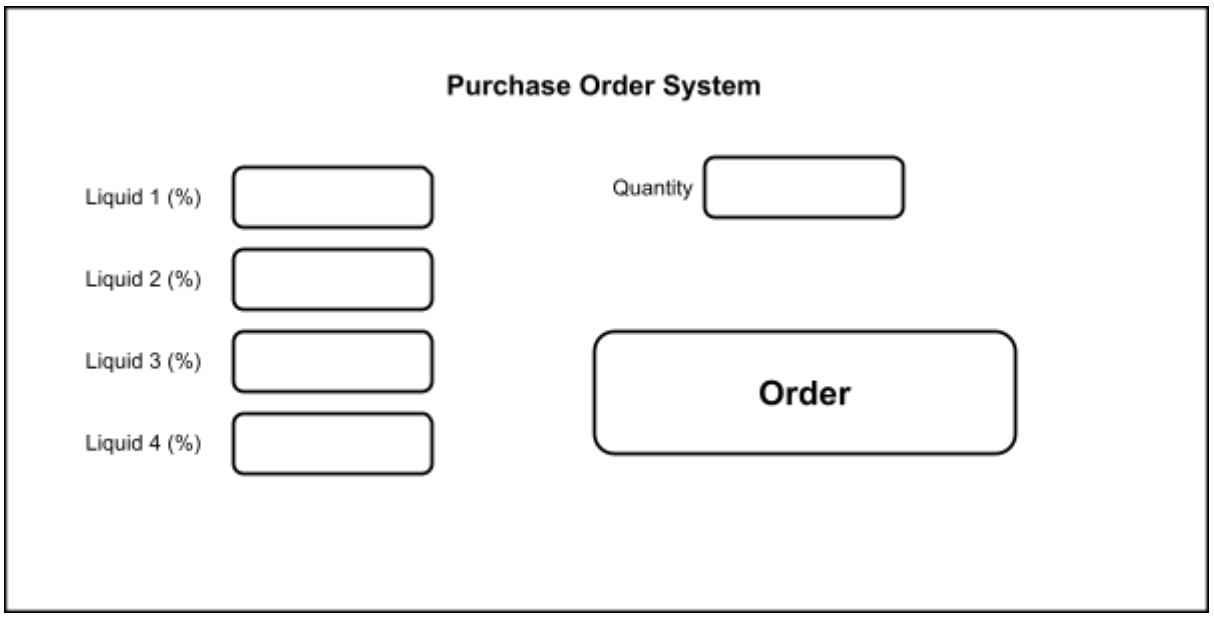
By Jouveer

Product/Purchase Order System (POS)

Brief:
Product order system (POS) allows the customers to launch orders directly on-line from their computers (for example, through a web-based application or some dedicated application). The order should contain information on liquid specification (selection of liquids that have to be mixed into each produced bottle) and number of bottles (quantity). The order should be delivered to the manufacturing system directly, processed and launched once it is scheduled for production. Design teams have freedom of specifying and implementing a meaningful POS that seamlessly integrates with ABS

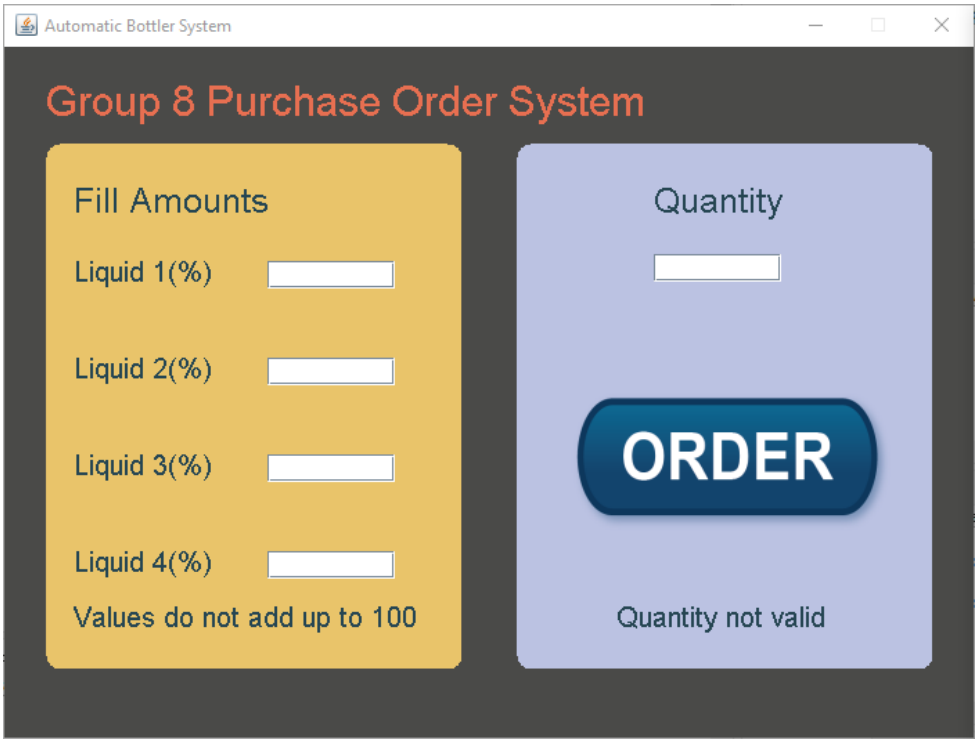
Original Design

The POS is used so that order can be placed, then given to the ABS. The ABS will take the information given by the user in the POS and schedule the manufacture of the required amount, and mix the liquids in the required proportions.
The user interface will probably be created in Java like, like the user interface given in the last lab.
A rough outline of the system can be seen below:

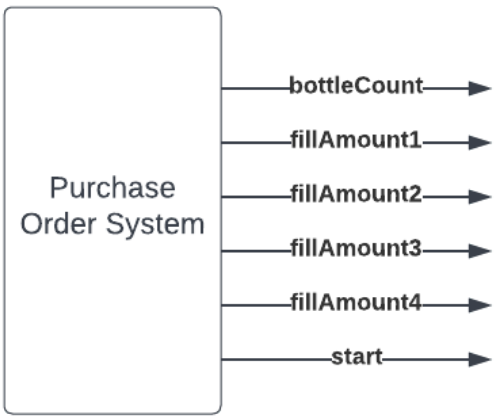


Final Implementation

Design:
The final implementation has a slightly different design than the original mock up. The colours that were decided upon were based on the colour scheme that was used on the GUI to represent what stage of the design was currently being run.



Signals:
The POS system was used as the main starting point for all the other systems. For anything in the design to start functioning properly, the start signal had to be sent from the POS system along with the pertaining information regarding the fill amounts for each liquid, and the amount of bottles that needed to be filled.
These data signals are sustained for the entire duration of the program execution. The system is designed to be run once, and then restarted if the system needs to accept another order.



Tools

For the GUI, the main tool that was used was Java's AWT and Swing libraries. This allowed the easy creation of the POS system using the same tools that the GUI for the ABS system was using.

```
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.setFont(g.getFont().deriveFont(30f));
        g.setColor(BURNTSIENNA);
        g.drawString("Group 8 Purchase Order System", 30, 50);
        g.setColor(MAIZECRAYOLA);
        g.fillRoundRect(30, 70, 300, 380, 20, 20);
        g.setColor(PERIWINKLECRAYOLA);
        g.fillRoundRect(370, 70, 300, 380, 20, 20);
        g.setColor(CHARCOAL);
        g.setFont(g.getFont().deriveFont(25f));
        g.drawString("Fill Amounts", 50, 120);
        g.setFont(g.getFont().deriveFont(20f));
        g.drawString("Liquid 1(%)", 50, 170);
        g.drawString("Liquid 2(%)", 50, 240);
        g.drawString("Liquid 3(%)", 50, 310);
        g.drawString("Liquid 4(%)", 50, 380);

        g.setFont(g.getFont().deriveFont(25f));
        g.drawString("Quantity", 469, 120);

        if (getTotal() != 100) {
            g.setFont(g.getFont().deriveFont(20f));
            g.drawString("Values do not add up to 100", 50, 420);
        }
        if (getQuantity() == 0) {
            g.setFont(g.getFont().deriveFont(20f));
            g.drawString("Quantity not valid", 442, 420);
        }
    }
```

Communication between POS and ABS system

```
try {
    filler1 = new SimpleClient(IP, 50000, "TopLevelCD", "fillAmount1E");
    filler2 = new SimpleClient(IP, 50000, "TopLevelCD", "fillAmount2E");
    filler3 = new SimpleClient(IP, 50000, "TopLevelCD", "fillAmount3E");
    filler4 = new SimpleClient(IP, 50000, "TopLevelCD", "fillAmount4E");
    bottleCount = new SimpleClient(IP, 50000, "TopLevelCD", "bottleCount");
    start = new SimpleClient(IP, 50000, "TopLevelCD", "start");
} catch (Exception e) {
    e.printStackTrace();
}
```

In order to facilitate the communication between the POS system and the ABS system that was required for sending information back and forth between them, the Sysj SimpleClient feature was used.

This allowed the system to emit, sustain and send data through signals properly in a way that SystemJ was expecting. One dedicated port was used for all POS GUI systems under the same clock domain. Signals could then be sent back and forth between the 2 systems.

Error checking and validation

Before an order was made, a few checks had to be done to ensure the proper operation of the automated bottling system.
Without these checks, the POS system would be able to send invalid information to the rest of the system like fill percentages that added up to over 100, or incorrect quantity numbers.
Before any start signals could be sent, these were checked and appropriate error messages were displayed

```
void sendOrder() {
    // Send order to the server
    System.out.println("Sending order");
    int quantity = getQuantity();
    int total = getTotal();
    if (quantity == 0 || total != 100) {
        System.out.println("Invalid order");
        return;
    }
    try {
        bottleCount.sustain(quantity);
        filler1.sustain(Integer.parseInt(liquid1.getText()));
        filler2.sustain(Integer.parseInt(liquid2.getText()));
        filler3.sustain(Integer.parseInt(liquid3.getText()));
        filler4.sustain(Integer.parseInt(liquid4.getText()));
        start.sustain(true);
    } catch (IOException e) {
        System.out.println("Order failed to send");
        e.printStackTrace();
    } catch (NumberFormatException e) {
        System.out.println("Invalid percentages");
        e.printStackTrace();
    }
}
```