A

Project

Based on

Wholesale System Management

Made By:

Dimple Nigah(C0705583)

Sarpreet Kaur(C0699659)

Ramanpreet Kaur(C0710904)

Shivaji Makkena(C0709432)

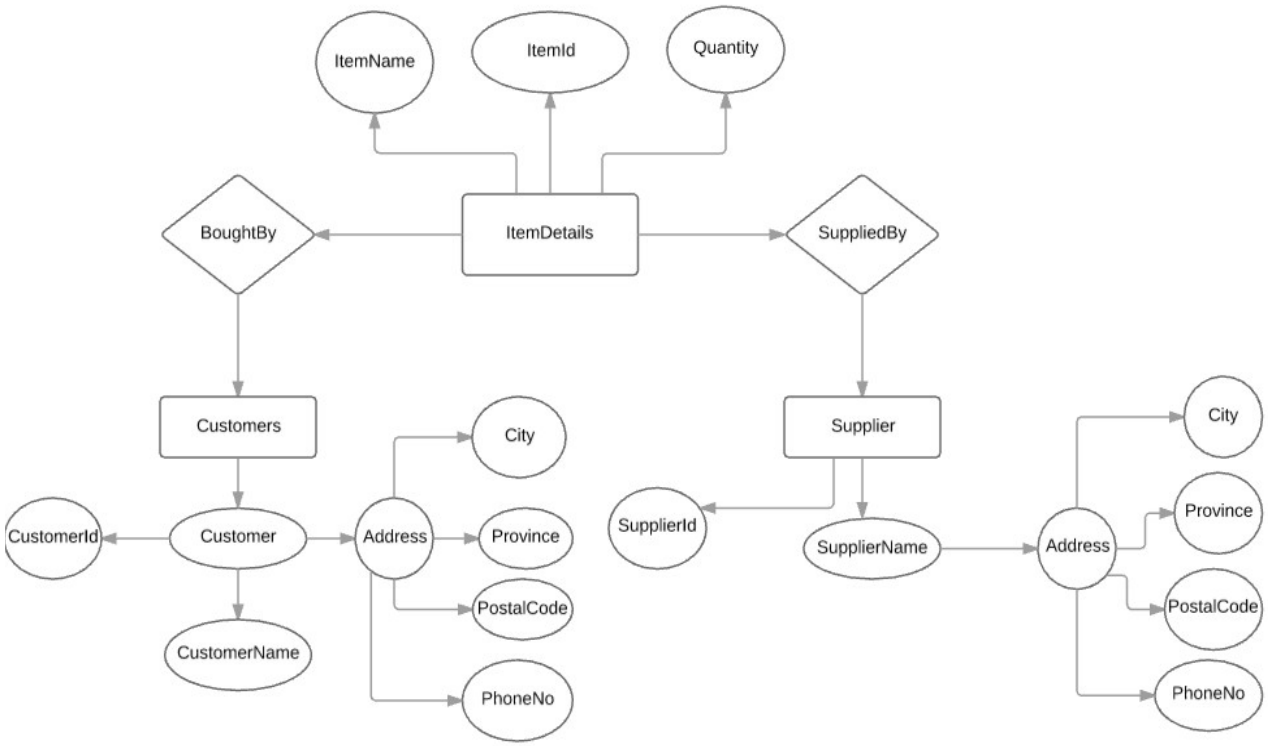
Submitted to :

Marcos BittenCourt

Project includes :

- Stock Details
- Customer Details
- Buyers Details
- Profit Calculation (Monthly)
- List of defaulters

ER-Diagram :

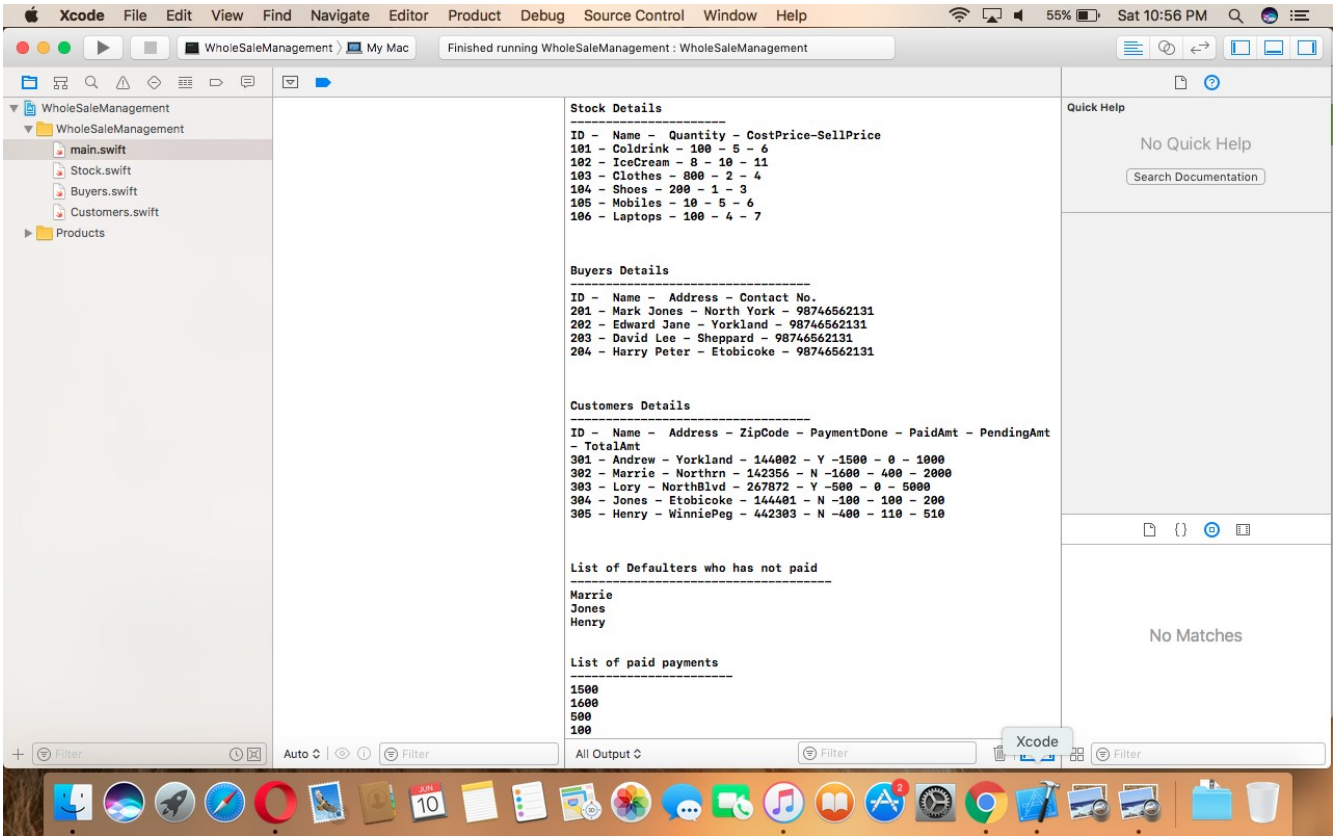


Qus.1 Stock Details?

```
class items
{
    var itemId : Int
    var itemName : String
    var quantity : Int
    var itemCostprice : Int
    var itemSellprice : Int
    //var profit : Double

    init(){
        itemId = 0
        itemName = ""
        quantity = 0
        itemCostprice = 0
        itemSellprice = 0
        //profit = 0.0
    }
    init(id : Int, n : String, qty : Int,cp : Int,sp : Int){
        self.itemId = id
        self.itemName = n
        self.quantity = qty
        itemCostprice = cp
        itemSellprice = sp
        //profit = p
    }

    func getitemId() -> Int {
        return itemId
    }

    func getitemName() -> String {
        return itemName
    }

    func getQuantity() -> Int {
        return quantity
    }
    func getcostprice() -> Int{
        return itemCostprice
    }
    func getsellprice() -> Int{
```

```
        return itemSellprice
    }
}
```

OutPut:

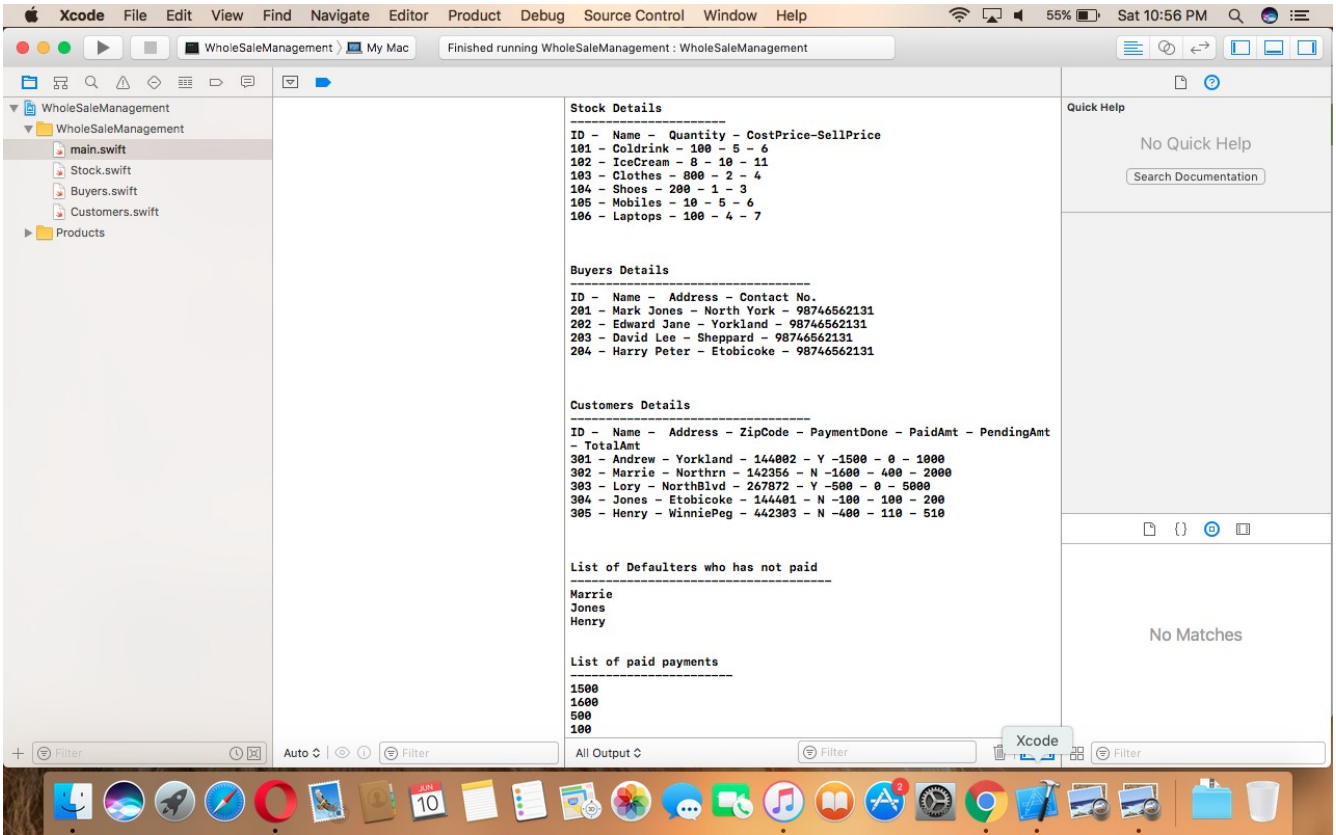


Qus 2. Buyers Details?

```
class buyers
{
    var buyerId : Int
    var buyerName : String
    var address : String
    var contact : Int
    /* init()
     {


     }*/
```

```
    init(bid : Int, bname : String, adrs : String,cntc : Int )
    {
        self.buyerId = bid
        self.buyerName = bname
        self.address = adrs
        self.contact = cntc
    }

    func getbuyerId() -> Int {
        return buyerId
    }

    func getbuyerName() -> String {
        return buyerName
    }

    func getaddress() -> String {
        return address
    }
    func getcntc() -> Int {
        return contact
    }
}
```

Output :

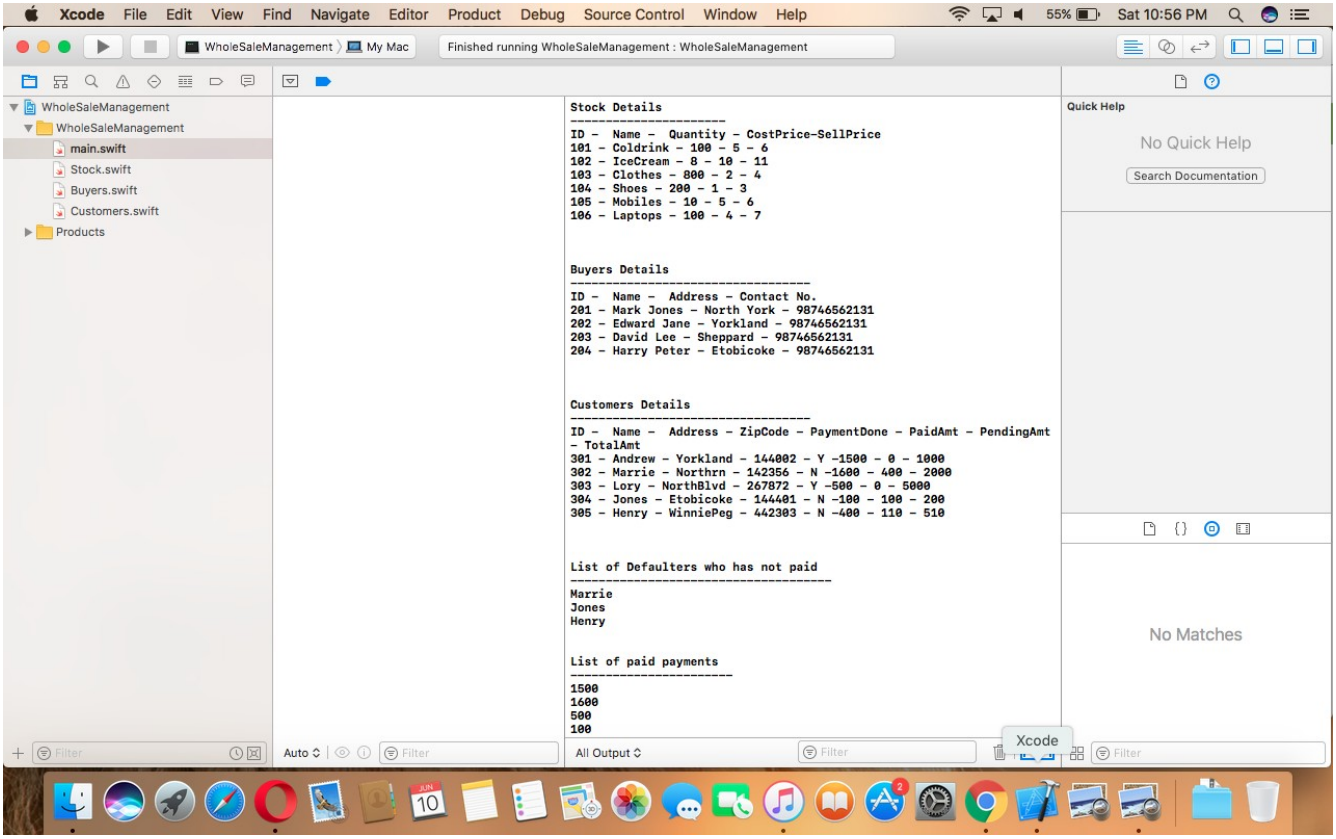


Qus 3.Customers Details?

```
class Customer
{
   var customerId : Int
   var customerName : String
   var customerAddress : String
   var zipCode : Int
   var payment : String
   var paidAmt : Int
   var pendingAmt : Int
   var totalAmt : Int
   init(cid : Int, cname : String, adrss : String,zip : Int, paymnt : String, paid : Int,pending : Int , total :
Int )
   {
      self.customerId = cid
      self.customerName = cname
      self.customerAddress = adrss
      self.zipCode = zip
      self.payment = paymnt
      self.paidAmt = paid
      self.pendingAmt = pending
      self.totalAmt = total
   }

   func getcustomerId() -> Int {
      return customerId
   }

   func getcustomerName() -> String {
      return customerName
   }

   func getcustomerAddress() -> String {
      return customerAddress
   }
   func getzipCode() -> Int {
      return zipCode
   }
   func getpayment() -> String{
      return payment
   }

   func getpaidAmt() -> Int{
```

```
        return paidAmt
    }
    func getpendingAmt() -> Int
    {
        return pendingAmt

    }
    func gettotalAmt()-> Int
    {
        return totalAmt
    }
}
```

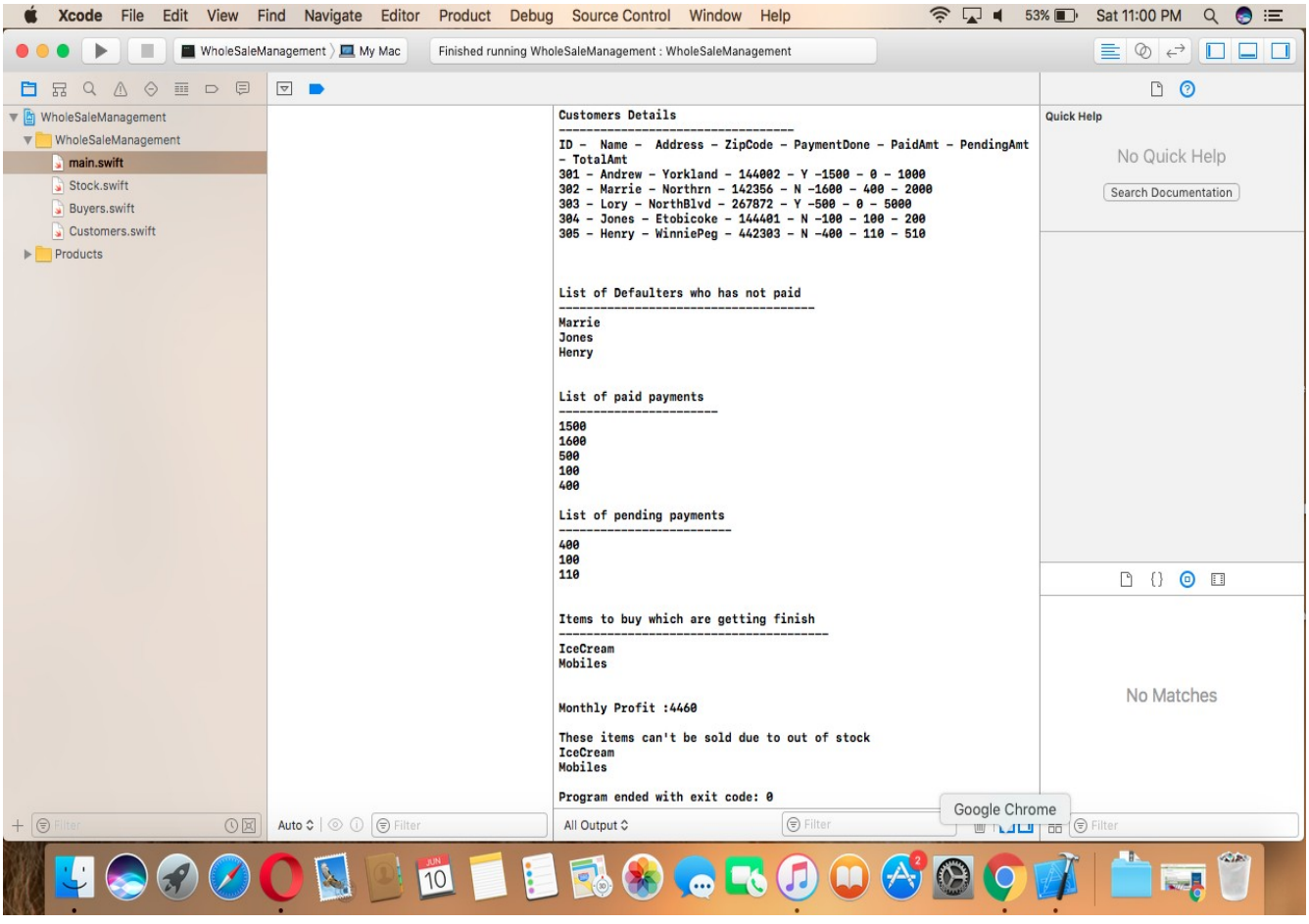
OutPut :

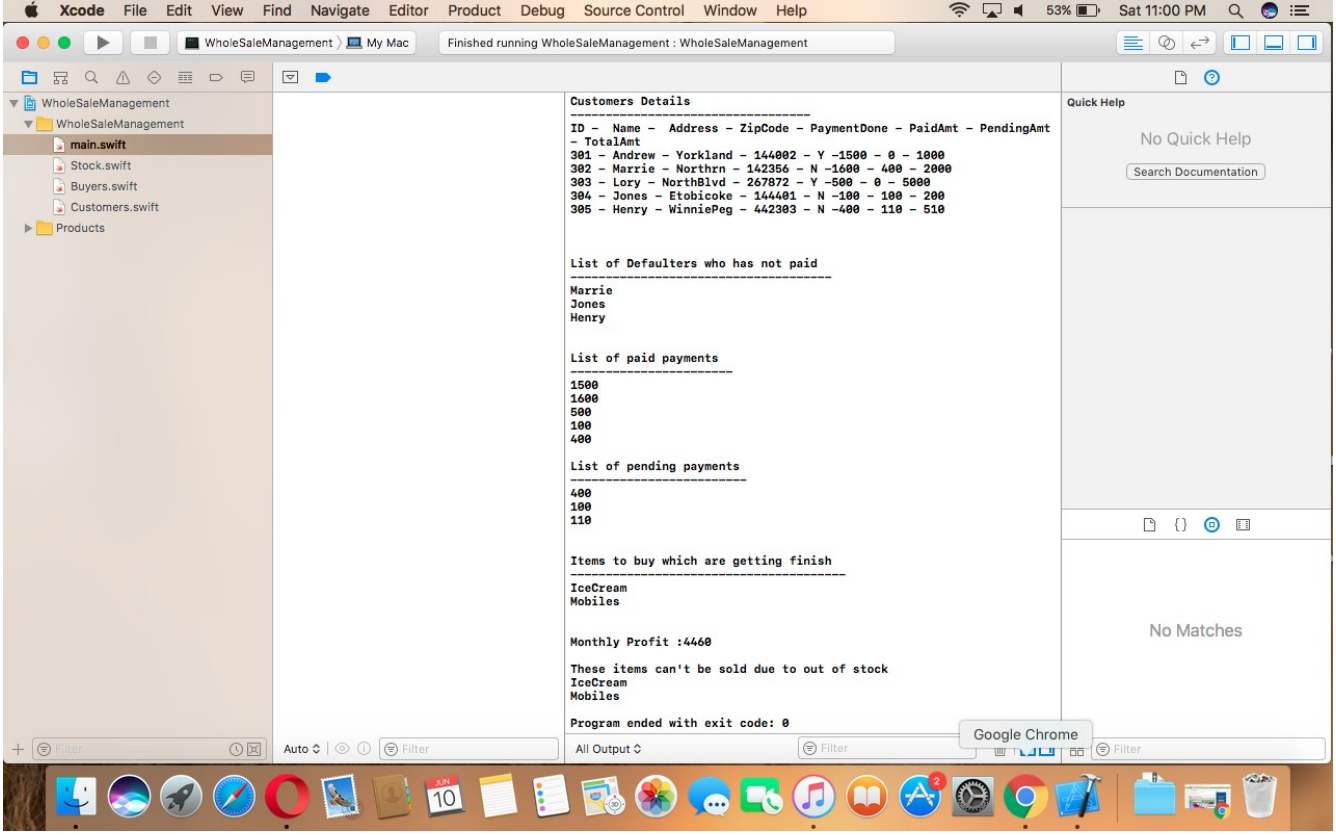


Qus 4. Defaulters who has not paid yet

```
//to show defaulters list
 func defaulterList() {

    print("\n\n\nList of Defaulters who has not paid")
    print("-----------------------------------")
    for x in custo{
      amtrcvd += x.getpaidAmt()
       if(x.getpayment() == "N"){
          print("\(x.getcustomerName())")
       }
    }
    print("")
 }
```

output:

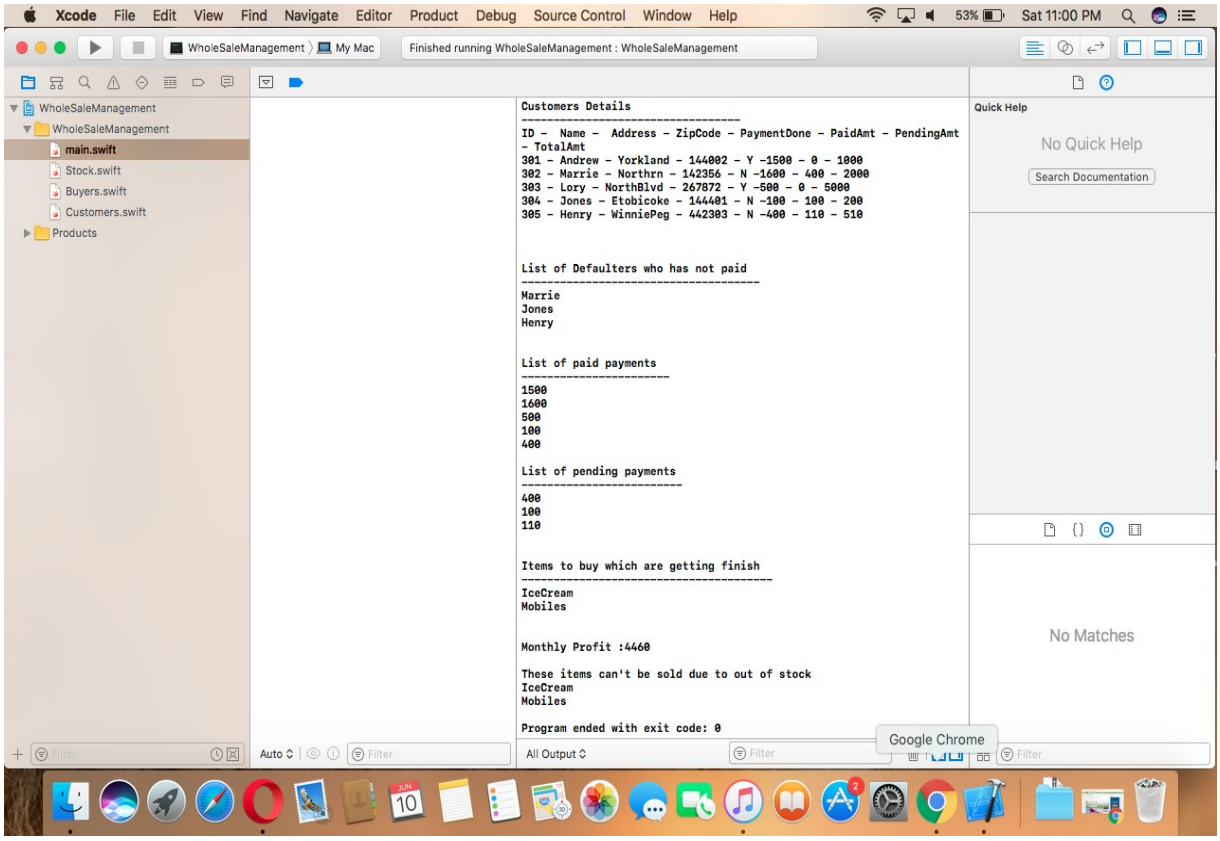




5. **List of payment paid and pending?**

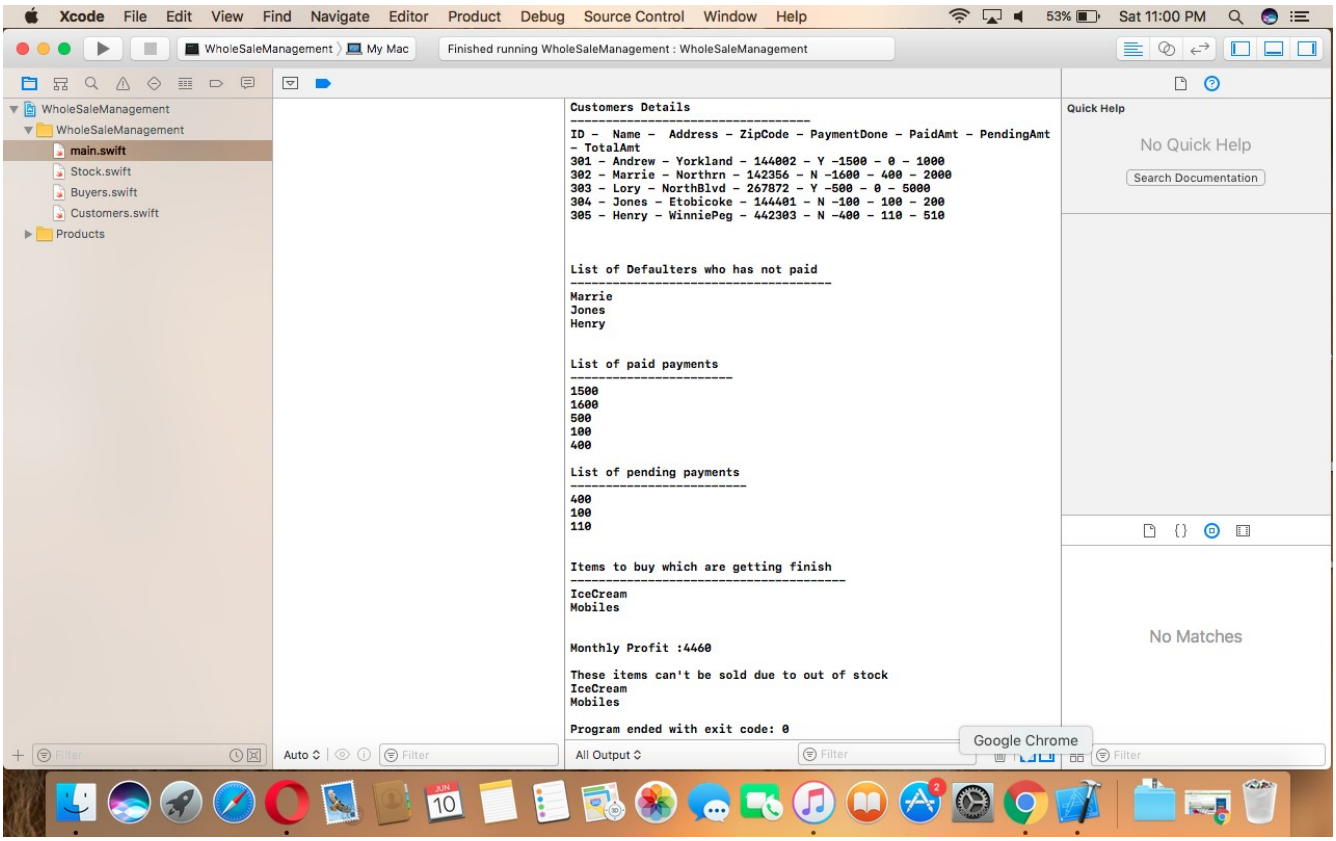
```
//To find total pending and paid
  func paidList() {
     //var paidlist = 0
     print("\nList of paid payments")
     print("----------------------")
     for x in custo{
           print("\(x.getpaidAmt())")

     }
     print("\nList of pending payments")
     print("-------------------------")
     for x in custo{
        if(x.getpendingAmt() > 0){
           print("\(x.getpendingAmt())")
        }
     }
     print("")
  }
```

OutPut :



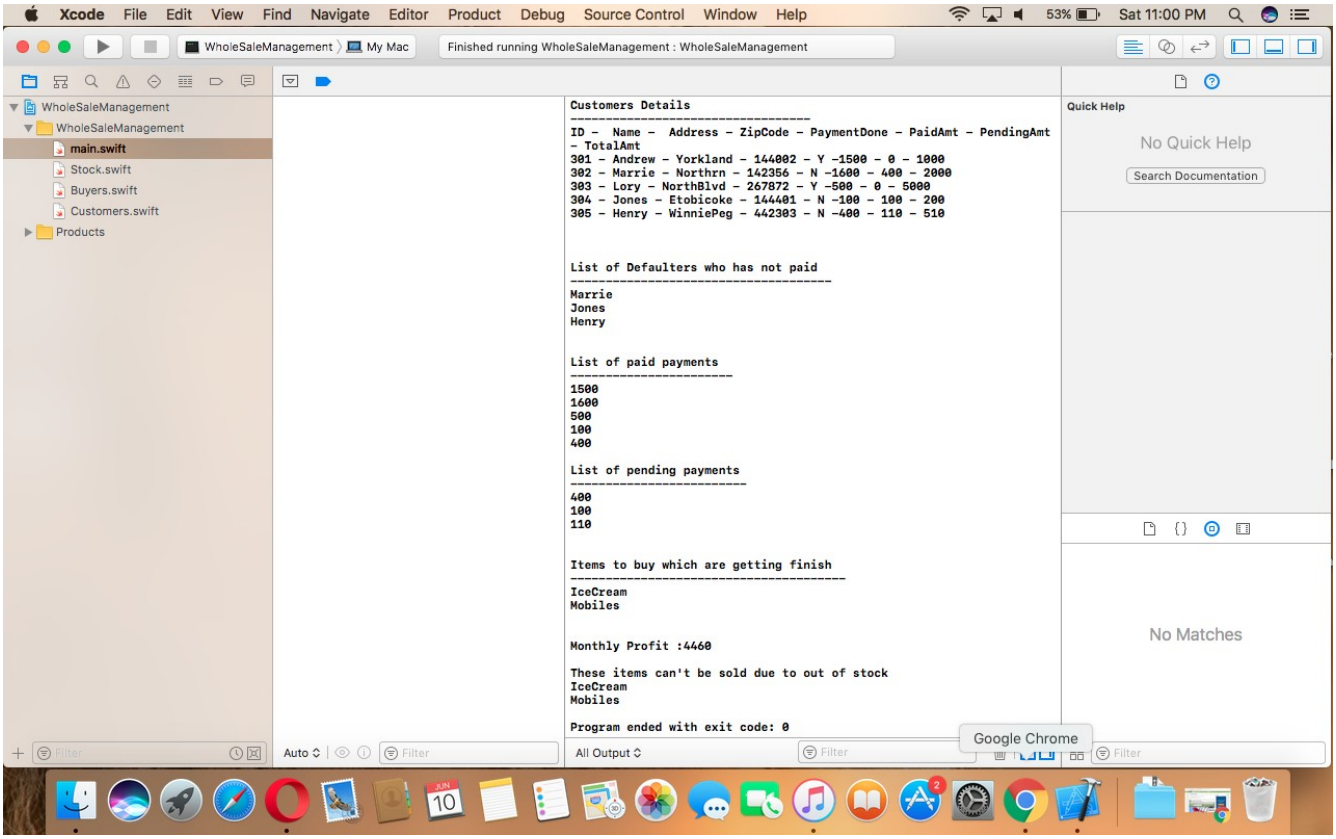
6. **Stock to buy that is less than particular amount ?**

```
//To find which items to buy if it's stock goes off
  func itemsTobuy(){
     print("\nItems to buy which are getting finish")
     print("-----------------------------------")
     for y in product{
        if(y.getQuantity() <= 10){
           print("\(y.getitemName())")
        }
     }
     print("")
  }
```



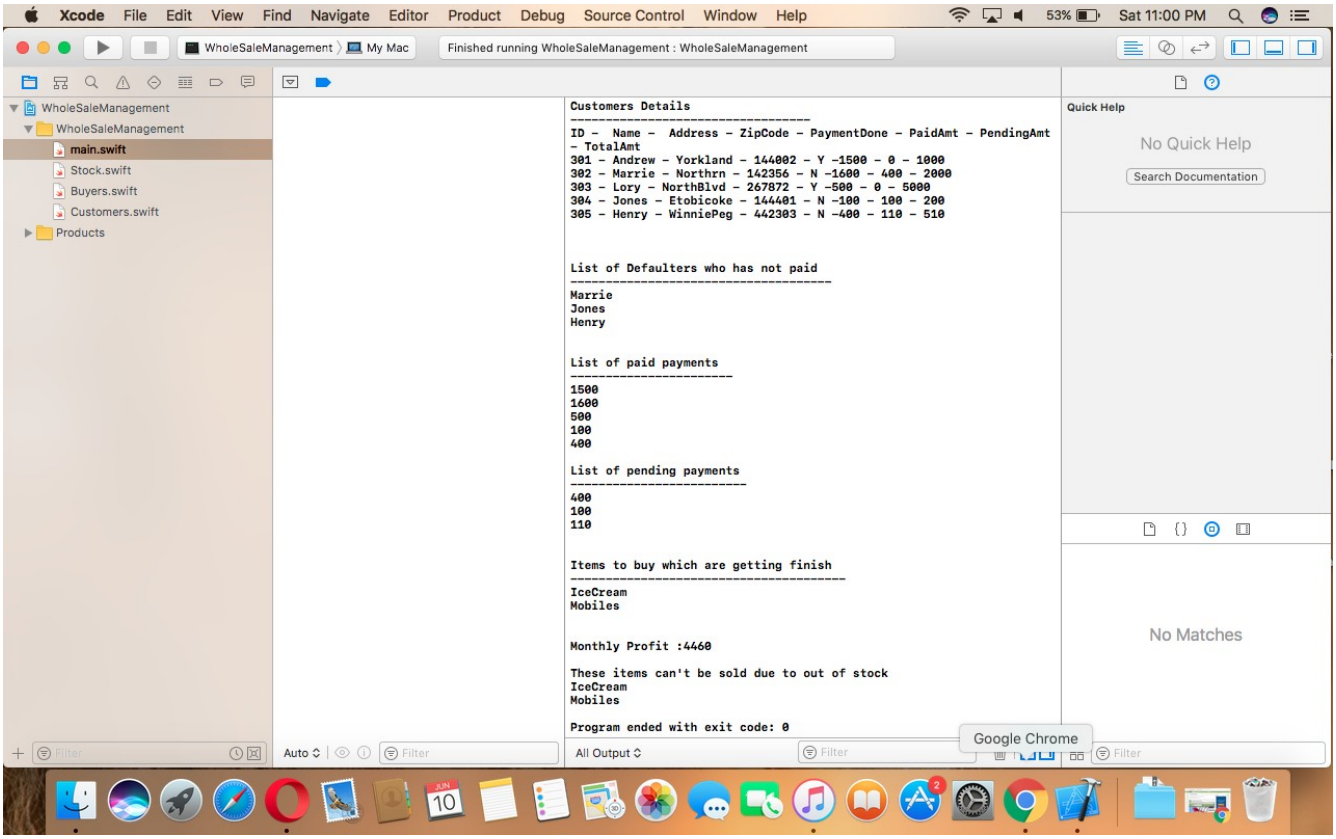
7. Profit Calculation Monthly

```
//To find Monthly Profit
  func profitCalculation()
  {
     var p : Int = 0
     var totalcp = 0
     var totalsp = 0

     for ss in product{
        if(ss.getcostprice() < ss.getsellprice())
        {
           totalcp += ss.itemCostprice * ss.getQuantity()
           totalsp += ss.itemSellprice * ss.getQuantity()
           p = p + (totalsp - totalcp)

        }
     }

     profit = p - amtrcvd
     print("\nMonthly Profit :\(profit)")
  }
```



8. Quantity not to sell if required amount is not present

```
//To Set Limit so that items can't be sold if these are out of stock
 func limit()
 {
    print("\nThese items can't be sold due to out of stock")
    for q in product{
       if(q.getQuantity() <= 10){
          print(q.getitemName())
       }
       else
       {
          product.append(items(id:q.getitemId(), n: q.getitemName(), qty: q.getQuantity() + 100,cp : 5, sp: 6))
       }
    }
    print("")
 }
```



Main File

```
class main
{

   var product : [items]
   var buyr : [buyers]
   var custo : [Customer]
   var amtrcvd : Int
   var profit : Int
   init()
   {
      product = []
      buyr = []
      custo = []
      amtrcvd = 0
      profit = 0
   }
}

var m = main();
```

```
m.addItems()
m.displayProducts()
m.addBuyers()
m.displayBuyers()
m.addCustomers()
m.displayCustomers()
m.defaulterList()
m.paidList()
m.itemsTobuy()
m.profitCalculation()
m.limit()
```