**Local cake shop has seen a recent surge in business and it has asked you to help design a system to manage the orders. The shop currently has the capacity to make up to 4 cakes per day and a cake can be stored up to 3 days prior to an event. To avoid last minute disasters, the shop will always make cakes at least 1 day prior to the event. The shop currently has the capacity to store up to 10 cakes. Standard policy is to block out 2 hours for delivery and setup at each event. The shop currently does not have the staff to handle more than one event at a time.**

**Design a system that will manage customer orders and event scheduling. Consider all of the factors listed in your design carefully. Sketch a user interface that will show all of the pertinent information to the cake shop employees. Design a flow chart and pseudocode that demonstrates how the system will manage the orders and inventory.**

**After completing the project, write an analysis of the process and identify any bottlenecks. If you were the cake shop owner, which bottleneck would you address first assuming price is not a factor? Explain your decision.**

**An analysis of the process and identify any bottlenecks. If you were the cake shop owner, which bottleneck would you address first assuming price is not a factor? Explain your decision.**

There is recent surge is business and orders cannot be managed manually due to the fact that there is lot of calculations/factors involved before confirming the order. For example:

* To check if there is no other order scheduled for delivery on the given date and time within 2 hours. If there is any other order scheduled for delivery within 2 hours, the order cannot be confirmed. This is one of the bottlenecks
* To calculate the preparation date. To calculate it, it is necessary to calculate the number of cakes to prepare 3 days prior to the delivery dates. The number of cakes that can be prepared in a day is 4. This is another bottleneck
* To check how many cakes are stored on a given date in shop? As maximum number of cakes that can be stored is 10 and cake can be prepared as there is availability and time available, cake cannot be prepared and stored. This is also a bottleneck for running the business

To overcome the issue of calculations and generating the order id, a system can be designed to handle these issues. It would be time saving as well as effective.

The bottleneck of not confirming the orders due to delivery, preparation and storing should be addressed first.

This design will keep asking to enter the delivery date and time and in parallel, it will keep on delivering the orders per the current date and time.

It will compare the given date and time and calculate if there is any date available up to three days prior to the delivery date. As maximum number of cakes, that can be prepared on any day, are four. If it can be prepared, the system will check if the order is deliverable. The delivery time should not conflict with any other delivery and there should be difference of 2 hours in each of the deliveries per the given constraint. If the cake is deliverable, it will check if there is place in shop to store the cake if prepared. The shop is limited to store only ten cakes.

In design, there are used different functions to calculate preparation date, if delivery is possible on given date and time, counting the number of cakes to prepare on any given date and how many cakes would be in shop before the preparation date.

There are loops in design to support the iteration over the orders and check the delivery dates and times. Also the status of delivered cakes should be stored to identify which orders have been delivered. This would help in determining the exact status of the shop.

The required input to the design is the delivery date and time as it is key factor to determine the preparation date. It will also be used to determine the preparation date. From preparation date, we can calculate the number of cakes present before the delivery date in the shop.

The required output would be the order number if the order is confirmed. If the order cannot be confirmed, the reason of the non confirmation of the order should be outputted.

This will help to identify the exact reason of refusing the order.

Pseudocode:

Start Main

//order number of the day

Declare orderNumber as Integer = 0

//to store list of date and time of delivery of each order

Declare deliveryDateAndTimes as Date List

//to store preparation date for the corresponding order

Declare preparationDates as Date List

Declare delivered as Boolean List

While true

Display “Enter dateTime of the delivery”

Input deliveryDateTime

preparationDate = getPreparationDate(deliveryDateAndTimes, orderNumber, deliveryDateTime, preparationDates)

if preparationDate = null

Display “Sorry, we don’t have time to prepare the cake for you”

Else

Count = getNumberOfCakesInShopOnGivenDate(deliveryDateAndTimes, orderNumber, preparationDate, delivered)

// **The shop currently has the capacity to store up to 10 cakes**

If count >= 10

Display “Sorry, we cannot store more than 10 cakes in shop”

Else

If isDeliveryPossible(deliveryDateAndTimes, orderNumber, deliveryDateTime)

//store the order

orderNumber = orderNumber + 1

deliveryDateAndTimes(orderNumber) = deliveryDateTime

preparationDates(orderNumber)= preparationDate

delivered(orderNumber) = false

Display “Thank you for the order. You order# is: “,orderNumber

Else

Display “Delivery is not possible on given date and time”

End if

End if

End if

deliverCakes(deliveryDateAndTimes, orderNumber, delivered)

End while

End Main

// **Standard policy is to block out 2 hours for delivery**

Declare isDeliveryPossible(deliveryDateAndTimes as Date List, orderNumber as Integer, deliveryDate as Date) as Boolean

For index = 1 to orderNumber

If deliveryDateAndTimes(index) = deliveryDate and absolute(deliveryDateAndTimes(index).time – deliveryDate.time)<2

Return false

End if

End for

Return true

End isDeliveryPossible

**// it will return preparation date**

Declare getPreparationDate(deliveryDateAndTimes as Date List, max as Integer, deliveryDate as Date, preparationDates as Date List) as integer

Declare count as Integer =0

Declare now as Date

//if needs delivery today only, not possible

If deliveryDate = now

Return null

End If

//if first order, make the cake previous day

If max = 0

Return deliveryDate-1

End if

For index = 1 to max

If deliveryDateAndTimes(index) = deliveryDate and deliveryDateAndTimes(index) – 1 IS NOT empty

For day = 1 to 3

Count = countNumberOfCakesToPrepareOnGivenDate(preparationDates, deliveryDate-day,max)

// **capacity to make up to 4 cakes per day**

If count < 4

Return deliveryDate-day

End if

End for

End if

End For

**//cannot prepare the cake upto three days before the event**

Return null

End getPreparationDate

**//to count number of cakes to prepare on the given day**

Declare countNumberOfCakesToPrepareOnGivenDate(preparationDates as Date List, proposedPreparationDate as Date, max as Integer) as Integer

Declare count as Integer =0

For index = 1 to max

If preparationDates (index) = proposedPreparationDate

count = count + 1

End if

End for

Return count

End countNumberOfCakesToPrepareOnGivenDate

**//to get number of cakes in store before delivery date**

Declare getNumberOfCakesInShopOnGivenDate(deliveryDateAndTimes as Date List, max as Integer, dateInQuestion as Date, delivered as Boolean List) as integer

Declare count as Integer =0

For index = 1 to max

If delivered(index)=false and dateInQuestion < deliveryDateAndTimes(index)

Count = count + 1

End if

End For

Return count

End getNumberOfCakesInShopOnGivenDate

**//deliver cake if it is the time**

Declare deliverCakes(deliveryDateAndTimes as Date List, high as Integer, delivered as Boolean List)

Declare now as Date

For index = 1 to max

If absolute(deliveryDateAndTimes(index).time – deliveryDate.time)>2

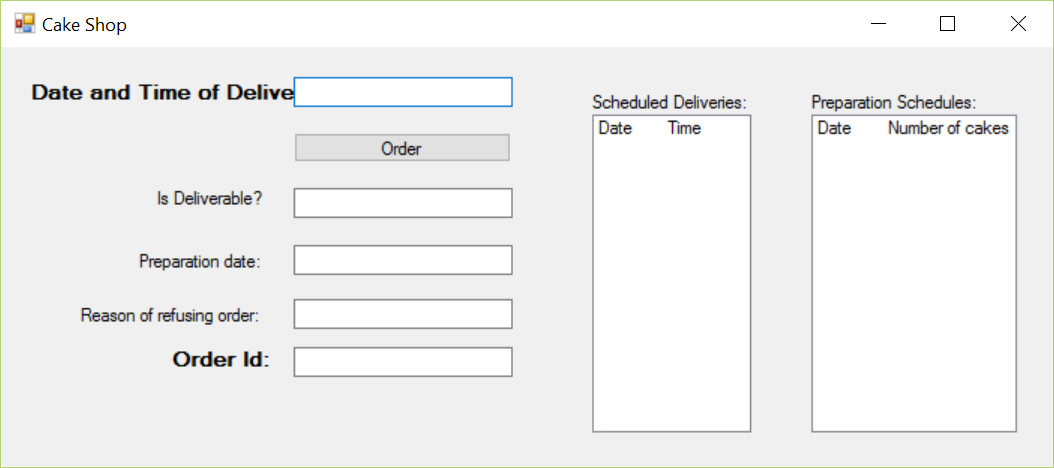
delivered(index) = true

End if

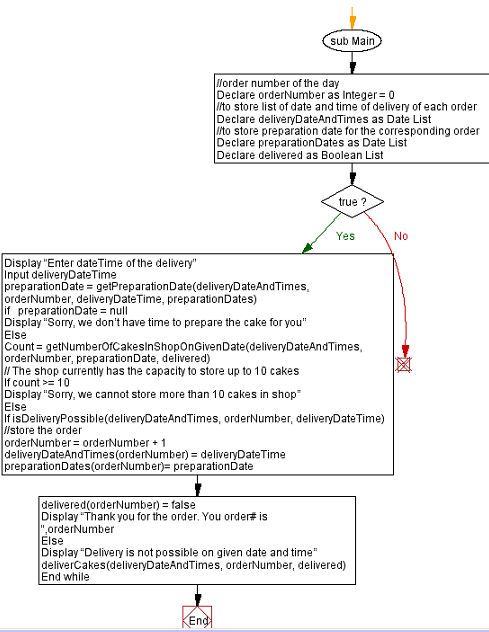
End For

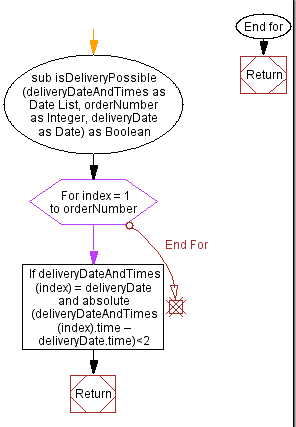
End deliverCakes

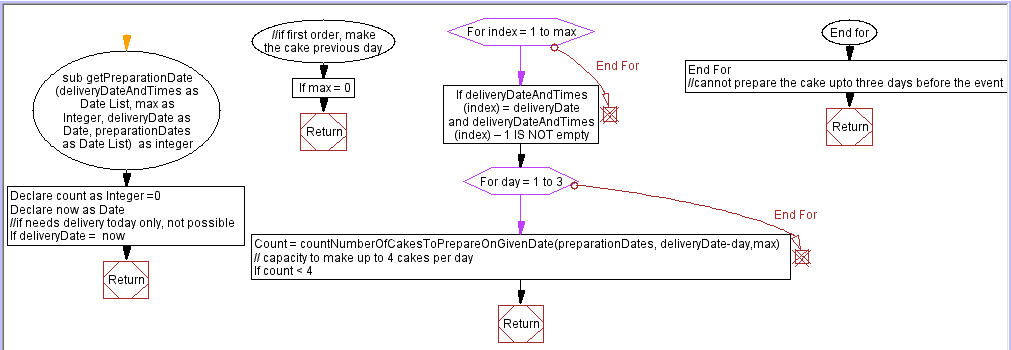
**A user interface that will show all of the pertinent information to the cake shop employees:**

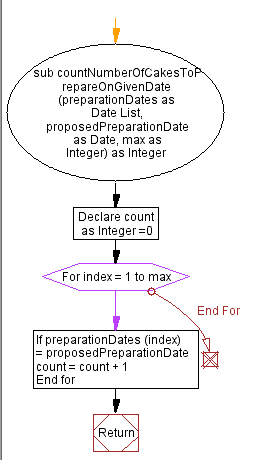


**Flowchart:**

****

****

****

****

