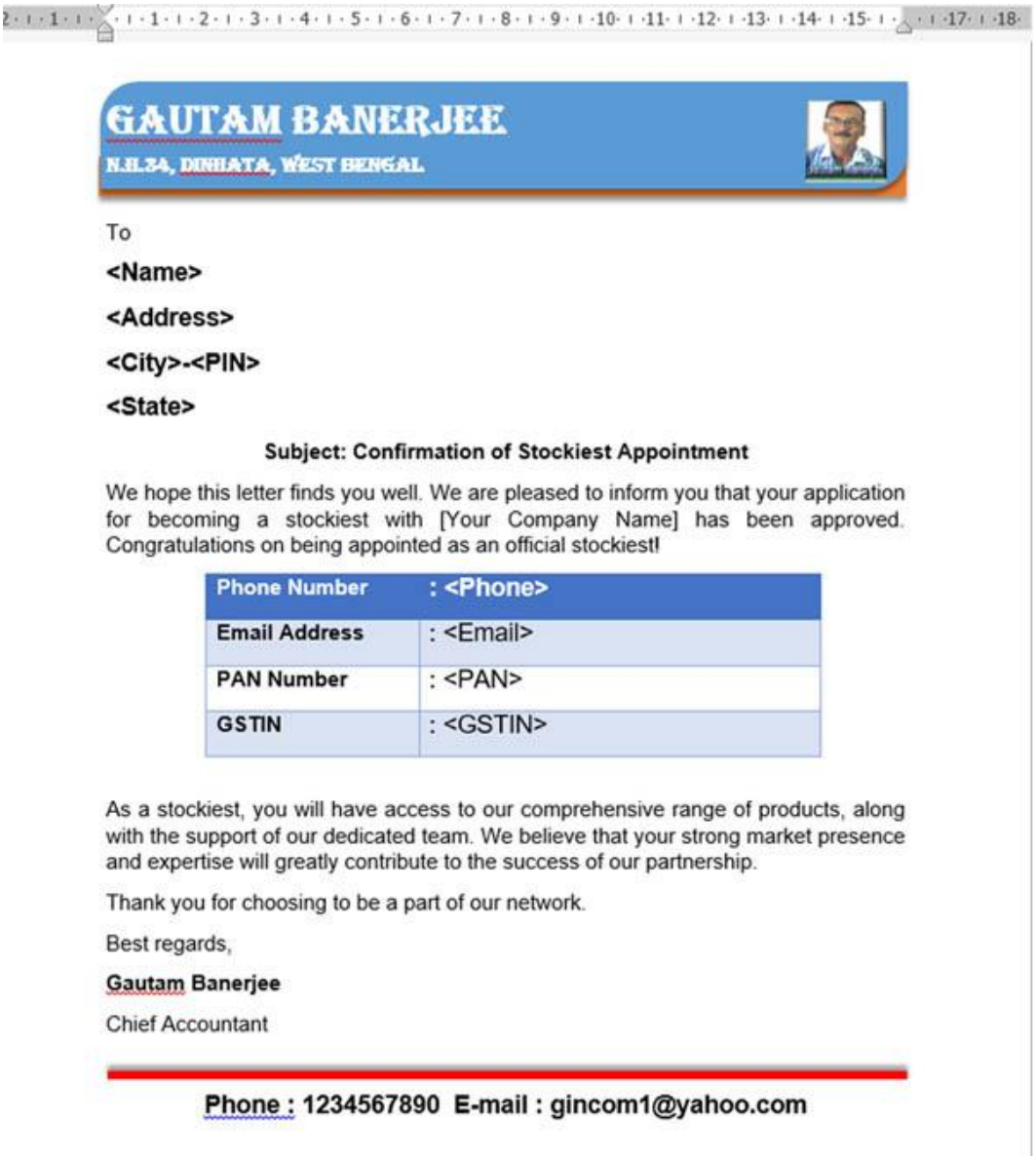
Effortless Customer Data Printing: Automate Your Workflow



GAUTAM BANERJEE

N.H.34, DINHATA, WEST BENGAL

To

<Name>

<Address>

<City>-<PIN>

<State>

Subject: Confirmation of Stockiest Appointment

We hope this letter finds you well. We are pleased to inform you that your application for becoming a stockiest with [Your Company Name] has been approved. Congratulations on being appointed as an official stockiest!

Phone Number	: <Phone>
Email Address	: <Email>
PAN Number	: <PAN>
GSTIN	: <GSTIN>

As a stockiest, you will have access to our comprehensive range of products, along with the support of our dedicated team. We believe that your strong market presence and expertise will greatly contribute to the success of our partnership.

Thank you for choosing to be a part of our network.

Best regards,

Gautam Banerjee

Chief Accountant

Phone : 1234567890 E-mail : gincom1@yahoo.com

The Automate process of sending personalized letters to your customers using the data from your "Customer_Master" records. Here's how you can approach this:

Step 1: Create a Word Document Template

Open Microsoft Word and create a template for your letter. Include placeholders such as <CustomerName>, <CustomerAddress>, etc. where you want to insert customer-specific information.

Step 2: Write VBA Code in Excel

Open the Excel workbook containing your "Customer_Master" records.

Press ALT + F11 to open the VBA Editor.

Insert a new module and write the VBA code for generating and sending letters.

Step 3: Read Customer Data

Use VBA to read data from your "Customer_Master" sheet. Loop through each customer's record and retrieve their name, address, and any other information you want to include in the letter.

Step 4: Open Word Document and Replace Placeholders

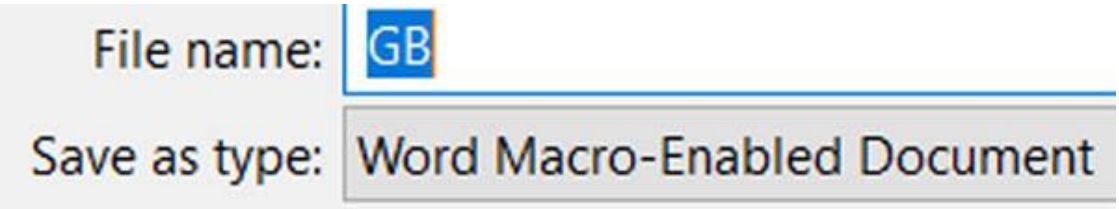
Use VBA to open the Word document template you created in Step 1.

Replace the placeholders in the Word document with the actual customer data using the Replace method.

Step 5: Save and Send Letters

Save the modified Word document with a unique name for each customer.

If you want to send the letters as attachments via email, you can use VBA's Outlook.Application to create emails with the Word documents as attachments.



Copy and Paste the Below Code:

Remember to adapt the code to match your file paths, sheet names, template, and other specific details. This is just a basic outline, and you might need to add error handling and additional functionality based on your requirements.

```
Private Sub PrntAck_Click()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Customer_Master")

    Dim WordApp As Object
    Dim WordDoc As Object
    Dim CustName As String
    Dim CustAddress As String
    Dim CustCity As String
    Dim CustPin As String
    Dim CustState As String
    Dim CustPhone As String
    Dim CustEmail As String
    Dim CustPAN As String
    Dim CustGSTIN As String

    ' Start Word application
    Set WordApp = CreateObject("Word.Application")
```

```
    WordApp.Visible = True

    ' Loop through each customer record
     For i = 2 To ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

        CustName = ws.Cells(i, 2).Value
        CustAddress = ws.Cells(i, 3).Value
        CustCity = ws.Cells(i, 4).Value
        CustPin = ws.Cells(i, 5).Value
        CustState = ws.Cells(i, 6).Value
        CustPhone = ws.Cells(i, 7).Value
        CustEmail = ws.Cells(i, 8).Value
        CustPAN = ws.Cells(i, 9).Value
        CustGSTIN = ws.Cells(i, 11).Value

        ' Open the Word template
        Set WordDoc =
WordApp.Documents.Open("C:\Users\user\Documents\GB.docm")
        'Set WordDoc =
WordApp.Documents.Open("C:\Users\user\Documents\Custom Office
Templates\AckTemplate.dot")
        'Set WordDoc =
WordApp.Documents.Open("E:\AckTemplate.docx")

        ' Replace placeholders with customer data
        WordDoc.Content.Find.Execute FindText:="<Name>",
ReplaceWith:=CustName
```

```
        WordDoc.Content.Find.Execute FindText:="<Address>",
ReplaceWith:=CustAddress
        WordDoc.Content.Find.Execute FindText:="<City>",
ReplaceWith:=CustCity
        WordDoc.Content.Find.Execute FindText:="<pin>",
ReplaceWith:=CustPin
        WordDoc.Content.Find.Execute FindText:="<State>",
ReplaceWith:=CustState
        WordDoc.Content.Find.Execute FindText:="<phone>",
ReplaceWith:=CustPhone
        WordDoc.Content.Find.Execute FindText:="<Email>",
ReplaceWith:=CustEmail
        WordDoc.Content.Find.Execute FindText:="<PAN>",
ReplaceWith:=CustPAN
        WordDoc.Content.Find.Execute FindText:="<GSTIN>",
ReplaceWith:=CustGSTIN


        ' Save and close the modified document
        WordDoc.SaveAs
"C:\Users\user\Documents\Ack_letter\AckTemplate_" & CustName &
".dot"
        WordDoc.Close
 Next i


    ' Clean up Word application
    WordApp.Quit
    Set WordDoc = Nothing
    Set WordApp = Nothing
End Sub
```

Line by Line Description:

```
Dim ws As Worksheet
Set ws = ThisWorkbook.Worksheets("Customer_Master")
```

In this code snippet, you are declaring a variable named ws of type Worksheet. You then use the Set keyword to assign an actual worksheet object to the ws variable. The worksheet being assigned is the one named "Customer_Master" within the workbook where the VBA code is running. This allows you to interact with and manipulate the data in the "Customer_Master" worksheet.

```
Dim WordApp As Object
Dim WordDoc As Object
Dim CustName As String
Dim CustAddress As String
Dim CustCity As String
```

In this part of the code, you are declaring several variables that will be used for interacting with Microsoft Word's object model.

WordApp is declared as an object that will represent the instance of Microsoft Word application. It's used to control Word and perform various tasks.

WordDoc is declared as an object that will represent a Word document. It will be used to open, manipulate, and save Word documents.

CustName and CustAddress are declared as strings. These variables will be used to store the customer's name and address retrieved from the Excel worksheet.

These variable declarations set the stage for interacting with both Excel (using the ws worksheet object) and Microsoft Word (using the WordApp and WordDoc objects) to generate customized letters.

```
Set WordApp = CreateObject("Word.Application")
WordApp.Visible = True
```

You are creating an instance of Microsoft Word's application using the CreateObject function with the argument "Word.Application". This essentially opens a new instance of Microsoft Word on your computer, which you can interact with using VBA code.

The second line, WordApp.Visible = True, sets the visibility of the Word application to "True", meaning that the Word application window will be shown on the screen. This allows you to see the Word documents being manipulated by the VBA code.

Overall, these lines of code create a connection to Microsoft Word and make it visible so that you can work with Word documents programmatically through VBA.

```
For i = 2 To ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

    CustName = ws.Cells(i, 2).Value
    CustAddress = ws.Cells(i, 3).Value
    CustCity = ws.Cells(i, 4).Value
```

A loop is being used to iterate through a range of cells in the "Customer_Master" worksheet (ws). Specifically, it's starting from the second

row (assuming the first row contains headers) and continues until the last non-empty cell in column A. Here's a breakdown of what's happening:

For i = 2 To ws.Cells(ws.Rows.Count, "A").End(xlUp).Row: This sets up a loop that will iterate from row 2 to the last non-empty row in column A of the "Customer_Master" worksheet. It's using the Rows.Count property to get the total number of rows in the worksheet and the End(xlUp).Row part to find the last non-empty row in column A.

Inside the loop, the values of the columns for the current row are being assigned to variables:

CustName = ws.Cells(i, 2).Value: This assigns the value in column 2 (B) of the current row to the CustName variable.

CustAddress = ws.Cells(i, 3).Value: This assigns the value in column 3 (C) of the current row to the CustAddress variable.

CustCity = ws.Cells(i, 4).Value: This assigns the value in column 4 (D) of the current row to the CustCity variable.

The loop allows you to process each row of data in the "Customer_Master" worksheet, extracting the values in columns 2, 3, and 4 for further use, possibly for generating letters or reports based on this data.

Set WordDoc = WordApp.Documents.Open("C:\Users\user\Documents\GB.docm")

WordDoc.Content.Find.Execute FindText:="<Name>", ReplaceWith:=CustName

WordDoc.Content.Find.Execute FindText:="<Address>", ReplaceWith:=CustAddress

WordDoc.Content.Find.Execute FindText:="<City>", ReplaceWith:=CustCity

The code is working with a Word document (WordDoc) that is being opened using the Open method of the Documents collection in the Word application. Here's what's happening step by step:

Set WordDoc = WordApp.Documents.Open("C:\Users\user\Documents\GB.docm"):

This line opens a Word document with the filename "GB.docm" located in the "C:\Users\user\Documents" folder. The Set keyword is used to assign the opened document to the WordDoc variable.

WordDoc.Content.Find.Execute FindText:="<Name>", ReplaceWith:=CustName:

This line uses the Find method to search for the placeholder text "<Name>" in the content of the opened Word document (WordDoc). When found, it's replaced with the value stored in the CustName variable.

WordDoc.Content.Find.Execute FindText:="<Address>", ReplaceWith:=CustAddress:

Similar to the previous line, this one searches for the placeholder text "<Address>" in the document's content and replaces it with the value stored in the CustAddress variable.

WordDoc.Content.Find.Execute FindText:="<City>", ReplaceWith:=CustCity:

This line searches for the placeholder text "<City>" and replaces it with the value stored in the CustCity variable.

The purpose of these lines is to find and replace specific placeholders within the Word document's content with the corresponding customer data obtained from the Excel worksheet. This allows you to personalize the content of the

WordDoc.SaveAs "C:\Users\user\Documents\Ack_letter\AckTemplate_" & CustName & ".dot"

WordDoc.Close

Next i

The code is saving the modified Word document with the replaced customer data to a specified folder. Here's what each line does:

WordDoc.SaveAs "C:\Users\user\Documents\Ack_letter\AckTemplate_" & CustName & ".dot":

This line uses the SaveAs method to save the modified Word document. It specifies the path where the document will be saved, along with the filename. The filename is constructed using the customer name (CustName) to make each document unique. The file extension ".dot" indicates a Word template file.

WordDoc.Close:

This line closes the current Word document that was opened and modified.

Next i: This line is part of a loop (For loop). It moves the loop to the next iteration, which involves processing the next customer's data and generating the corresponding letter.

In summary, these lines of code are responsible for saving the modified Word document with customer-specific data to a designated folder. The loop iterates through each customer's data, generating and saving personalized letters for each customer. The letters are saved as Word template files (with a ".dot" extension) in the "Ack_letter" folder.

WordApp.Quit

Set WordDoc = Nothing

Set WordApp = Nothing

WordApp.Quit: This line closes the Microsoft Word application that was opened for generating and modifying the letters. It ensures that the Word application is properly closed after the letters have been generated and saved.

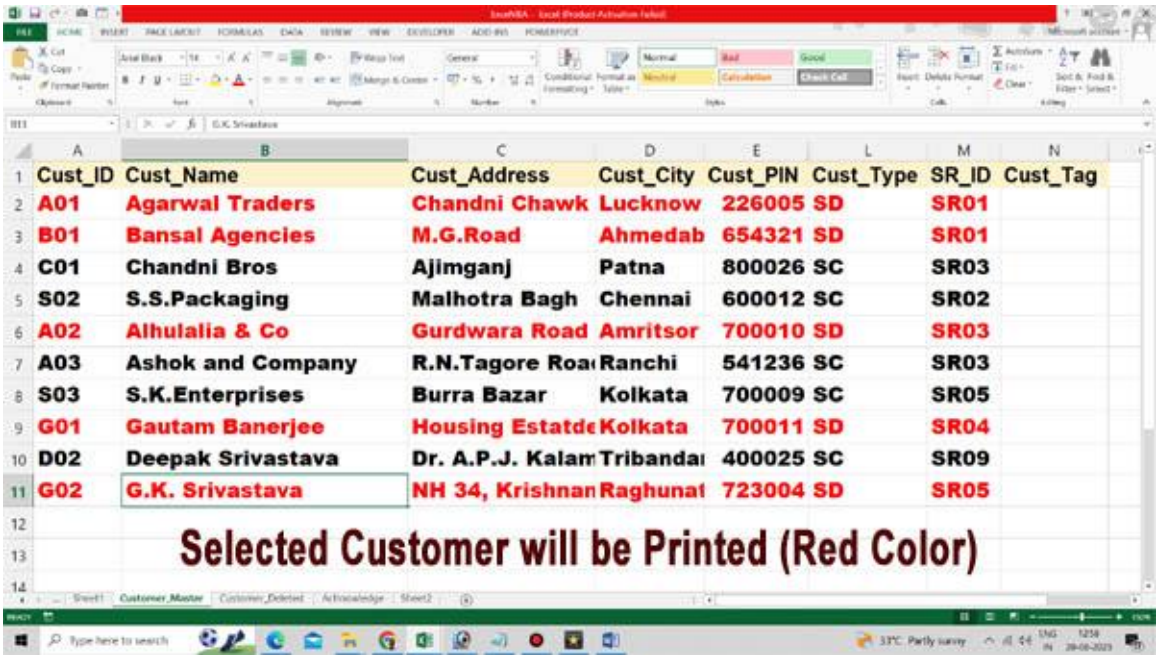
Set WordDoc = Nothing: This line releases the reference to the WordDoc object, effectively disconnecting it from the opened Word document.

Releasing the reference helps to free up memory and resources associated with the document.

Set WordApp = Nothing: This line releases the reference to the WordApp object, which represents the Microsoft Word application instance that was created. Releasing this reference ensures that the Word application instance is properly closed and its resources are released.

Overall, these lines of code ensure that any resources used by the Microsoft Word application and documents are properly released and cleaned up, preventing memory leaks and improving the efficiency of your VBA program.

Selective Customer Letter Print:



Cust_ID	Cust_Name	Cust_Address	Cust_City	Cust_PIN	Cust_Type	SR_ID	Cust_Tag
A01	Agarwal Traders	Chandni Chawk	Lucknow	226005	SD	SR01	
B01	Bansal Agencies	M.G.Road	Ahmedab	654321	SD	SR01	
C01	Chandni Bros	Ajimganj	Patna	800026	SC	SR03	
S02	S.S.Packaging	Malhotra Bagh	Chennai	600012	SC	SR02	
A02	Alhulalia & Co	Gurdwara Road	Amritsor	700010	SD	SR03	
A03	Ashok and Company	R.N.Tagore Roa	Ranchi	541236	SC	SR03	
S03	S.K.Enterprises	Burra Bazar	Kolkata	700009	SC	SR05	
G01	Gautam Banerjee	Housing Estatde	Kolkata	700011	SD	SR04	
D02	Deepak Srivastava	Dr. A.P.J. Kalam	Tribandai	400025	SC	SR09	
G02	G.K. Srivastava	NH 34, Krishnan	Raghunat	723004	SD	SR05	

Selected Customer will be Printed (Red Color)

```
For i = 2 To ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
    CustType = ws.Cells(i, 12).Value

    If CustType = "SD" Then
        ' The code inside this block will execute for customers with CustType = "SD"
        ' You can place your desired code here to perform specific actions for these customers
        ' For example, you can generate letters or perform other operations.
        ' ...
    End If

    ' The loop will continue to the next iteration to check the next customer
Next i
```

The lines of code you provided are inside a loop that iterates through each row of the "Customer_Master" worksheet to check the value in the "CustType" column. If the value in the "CustType" column is equal to "SD", then a block of code is executed. Here's a breakdown of how this works:

In this code, you're iterating through each row in the "Customer_Master" worksheet, and for each row, you're retrieving the value in the "CustType" column using ws.Cells(i, 12).Value. If the value is "SD", then the code block

within the If CustType = "SD" Then ... End If is executed, allowing you to perform specific actions for customers with CustType "SD". You can customize the code within the If block to suit your needs, such as generating letters or performing other tasks. The loop continues to the next iteration to process the next customer record.

Copy and Paste Full code for selected Customer Letter Print:

```
Private Sub PrntAck_Click()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Worksheets("Customer_Master")

    Dim WordApp As Object
    Dim WordDoc As Object
    Dim CustName As String
    Dim CustAddress As String
    Dim CustCity As String
    Dim CustPin As String
    Dim CustState As String
    Dim CustPhone As String
    Dim CustEmail As String
    Dim CustPAN As String
    Dim CustGSTIN As String
```

```
' Start Word application
Set WordApp = CreateObject("Word.Application")
WordApp.Visible = True

' Loop through each customer record
 'For i = 2 To ws.Cells(ws.Rows.Count, "A").End(xlUp).Row

For i = 2 To ws.Cells(ws.Rows.Count, "A").End(xlUp).Row
    CustType = ws.Cells(i, 12).Value

 If CustType = "SD" Then

    CustName = ws.Cells(i, 2).Value
    CustAddress = ws.Cells(i, 3).Value
    CustCity = ws.Cells(i, 4).Value
    CustPin = ws.Cells(i, 5).Value
    CustState = ws.Cells(i, 6).Value
    CustPhone = ws.Cells(i, 7).Value
    CustEmail = ws.Cells(i, 8).Value
    CustPAN = ws.Cells(i, 9).Value
    CustGSTIN = ws.Cells(i, 11).Value

    ' Open the Word template
```

```
    Set WordDoc = WordApp.Documents.Open("C:\Users\user\Documents\GB.docm")
    'Set WordDoc = WordApp.Documents.Open("C:\Users\user\Documents\Custom Office Templates\AckTemplate.dot")
    'Set WordDoc = WordApp.Documents.Open("E:\AckTemplate.docx")

    ' Replace placeholders with customer data
    WordDoc.Content.Find.Execute FindText:="<Name>", ReplaceWith:=CustName
    WordDoc.Content.Find.Execute FindText:="<Address>", ReplaceWith:=CustAddress
    WordDoc.Content.Find.Execute FindText:="<City>", ReplaceWith:=CustCity
    WordDoc.Content.Find.Execute FindText:="<pin>", ReplaceWith:=CustPin
    WordDoc.Content.Find.Execute FindText:="<State>", ReplaceWith:=CustState
    WordDoc.Content.Find.Execute FindText:="<phone>", ReplaceWith:=CustPhone
    WordDoc.Content.Find.Execute FindText:="<Email>", ReplaceWith:=CustEmail
    WordDoc.Content.Find.Execute FindText:="<PAN>", ReplaceWith:=CustPAN
```

```
        WordDoc.Content.Find.Execute FindText:="<GSTIN>", ReplaceWith:=CustGSTIN


        ' Save and close the modified document
        WordDoc.SaveAs "C:\Users\user\Documents\Ack_letter\AckTemplate_" & CustName & ".dot"
        WordDoc.Close
    End If
    Next i


    ' Clean up Word application
    WordApp.Quit
    Set WordDoc = Nothing
    Set WordApp = Nothing
End Sub
```

How to add comments in VBA coding.

In VBA, you can add comments to your code to provide explanations, descriptions, or notes that help you and others understand the code's purpose and functionality. Comments are ignored by the VBA compiler and are intended for human readers.

There are two types of comments in VBA:

Single-line Comments: These comments are used for a single line of text.

To add a single-line comment, use an apostrophe ' at the beginning of the line:

' This is a single-line comment

Multi-line Comments: These comments can span multiple lines and are enclosed within /* ... */.

To add a multi-line comment, enclose the text within /* and */:

/* This is a multi-line comment */

Here's an example of how comments can be used in VBA code:

Sub ExampleCode()

' This is a comment explaining the purpose of the code

Dim x As Integer x = 10

'This comment explains the value assignment

' Multi-line comment

/* The following loop iterates through an array and performs calculations on each element */

For i = 1 To 5 ' Calculate square of i square = i * i

Debug.Print "Square of " & i & " is " & square Next i

' End of the program

MsgBox "Code execution completed!" End Sub

Adding meaningful comments to your code can greatly improve its readability and maintainability, making it easier for you and others to understand and work with the code in the future.

Gautam Banerjee

“Helping beginners learn something new is a great way to share your knowledge and make a positive impact”.

Email: gincom1@yahoo.com

If you have any queries, please visit our "Contact Us" page.

Thank You. See you again!

PLEASE DONATE

Gautam Banerjee

Age: 63

Pay by UPI

9748327614