“VARDHAMAN TEXTILE”

A

PROJECT REPORT

***Submitted* By**

Group-1

CS11M020 – Dinesh Parmar

CS11M021 – Gaurav Gurav

CS11M041 – Preetham V

CS11M043 – Priyatosh Mishra

CS11M046 – Rishi Garg

CS11M050 – Saurav Kant Jha

CS11M051 – Shabbir Hussain

CS11M054 – Siddharth Kumar Jain

In

Software Engineering Theory and Practices

Under the guidance of

Prof. V. Kamakoti

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY

CERTIFICATE

This is to certify that the project entitled - "**Vardhaman Textile**" has been completed by Group 1 in the year 2011 under my guidance. This report has been submitted to the **Department of Computer Science and Engineering, INDIAN INSTITUTE OF TECHNOLOGY MADRAS** as part of the course "**Software Engineering Theory and Practice"** in M.Tech. First semester.

Date : ----------.

Prof. V . Kamakoti

CSE, IIT Madras

Team Description

Roll No.	**Name**
CS11M020	Dinesh Parmar
CS11M021	Gaurav Gurav
CS11M041	Preetham V
CS11M043	Priyatosh Mishra
CS11M046	Rishi Garg
CS11M050	Saurav Kant Jha
CS11M051	Shabbir Hussain
CS11M054	Siddharth Kumar Jain

Acknowledgement

The successful completion of the project has been the result of hard work and cooperation of not only the team members but also few other people, without whom this would not have been possible. First and foremost we wish to express our deep sense of gratitude to our project guide, **Prof. V. Kamakoti** for his continuous motivation and valuable help throughout the project work and TA’s for their support during the development of the project. Our sincere thanks to almighty God and our parents.

Preface

The project entitled “Vardhaman Textile” has been worked out with a view to provide an effective, fast, and reliable way to manage Business Process of “Vardhaman Textile” – a cloth merchant shop.

This project is basically a desktop form application whose motive is to provide user with an interface to effectively carry out business work like Billing, Receipt, Ledger, Account maintenance etc. This project is a live project and will be deployed in a server in the user’s office. This Software is very user friendly and can be used by any novice user familiar with the business.

Chapter 1

Introduction

1.1 Project overview

The objective of our project is to automate the accounting and billing process of ‘Vardhaman Textile’- a whole sale cloth merchant shop. The product intends to mimic the physical process of billing and accounting of the shop. Its interface should also provide the user additional features like easy editing, auto fill prompts, error checking, etc. Besides accounting and billing features it should have the functionality of maintaining accounts corresponding to each customer, supplier, bank and employee.

1.2 Scope of the project

‘Vardhaman Textiles’ uses traditional methods of hand written bills for billing, ledger book for accounting and other manual process for record keeping. The work involved in this is tedious for large number of transactions. Our product aims to move this work to the computer so that certain tasks can be automated to reduce the human work involved.

Chapter 2

The Team

2.1 Blue Team

Priyatosh Mishra

Siddharth Kumar Jain

2.2 Green Team

Preetham V

Rishi Garg

Saurav Kant Jha

2.3 Yellow Team

Dinesh Parmar

Gaurav Gurav

Shabbir Hussain

Chapter 3

Epic

3.1 Overall description of the project

It is a business project for accounting of sale and purchase from the prospective of a wholesale cloth merchant.

We buy product from supplier either directly or indirectly through the broker. Broker comes to our shop with his product samples. Whenever we buy product through broker, they give us invoice estimate where details of products are written, some terms and conditions of goods are negotiated and means of transport with specification (transport name) are also discussed. When all these things are finalised, this invoice is discussed with the supplier and expected day of delivery is fixed. Generally, this time should be minimum but in some cases this time is specified differently by us.(- like In the case of winter item whose season are near to Deepavali (October - November) but whose display by broker starts from august itself. We make the order in august and decide to get delivery on a later date). Once this is done from the tentative invoice, broker creates actual invoice and send it to us by a post, in the meantime this order is also been posted to the supplier.

Once bill is received we may make payment or may decide to make the payment later. Irrespective of payment, goods are sent by some means of transport (called as garage) and we receive one garage slip in which various details of parcel (known as bales) are provided. Later we receive goods by these means of transport. During this process we make payment for goods with various additional charges (broker commission, Goods state entry tax, Garage transportation charge …) and considering all these charges and our profit, we calculate our goods rate (selling price). As told earlier that payment to supplier can be made whenever we receive bill from him but this is not done in most cases. In cases where payment is delayed, according to date

difference and terms settled earlier, some extra amount as interest is to be paid to the broker or supplier.

The goods are then sold at the newly calculated rate. Our customers are mostly fixed, meaning once any retailer or wholesaler comes to our shop then he is most likely to come again. Depending upon the customer's linkage with other shops and taking some reference we allow a customer to delay his payments for some time span. Whenever customer comes to the shop, he makes payment of previous bills, takes new goods and defers payment, or takes some goods and makes cash payment.

The concept of delayed payment gives us the freedom to make payment whenever we have cash. Whenever we make delayed payment after specified period of time, then some extra amount known as expenses and / or interest is charged. Delayed payment is encouraged because this helps us in gaining extra profit with respect to goods being sold. We also have the freedom of advance payment according to we which we are liable to get extra discount from supplier, based on some agreement. All these are also applicable when customers make payment to us. To keep track of these payments, we have to maintain accounts of suppliers and customers.

We maintain record for each supplier and customer. It is known as ledger. This keeps track of all transaction for each customer/supplier payment. Payments can be made through bank by DD or Cheque or directly through Cash. We transfer all cash received from customers in to bank. All these transactions are mentioned in the ledger.

The software being developed must handle the Billing and accounting in all above cases. Although there are various more terminology associated with these things, but our software for now is restricted only to these terms which can later be expanded to also include things like income tax , daily reports , profit and loss account , liabilities , expenses , cash on hand calculation etc.

Chapter 4

Speed Boat

4.1Engines

- One team member, Siddharth, is familiar with the manual process being automated.
- Tasks being computerized are mathematical and repetitive in nature, which can easily be performed in computer.

4.2 Anchors

- Most team members do not know much about the business process of a cloth merchant shop.
- The users of product are not used to working on a computer.
- The users of the product are specific about the process being followed and don’t want to change most of the methods they work with.
- Process being modelled is critical. Any form of mistake can lead to financial losses.

Chapter 5

Product Tree

5.1 Must Have

- Account Maintenance
- Billing Interface
- Price List
- Bill Printing
- Receipt
- Bill Payment Modes
- Ledger
- Cheque Bounce Entry
- Backup
- Restore

5.2 Can Have

- Billing RG Entry
- Summary Report
- Supplier Payment
- Supplier Receipt
- Garage Entry
- Other Reports

5.3 Nice to Have

- Employee Attendance
- Manual Bill Entry
- Phone Directory

Chapter 6

Stories

6.1 Story titles

- ID-464: Account Maintenance
- ID-517: Billing Header
- ID-518: Billing Body
- ID-519: Billing Footer
- ID-520: Billing RG Entry
- ID-465: Price List
- ID-535: Bill Printing
- ID-538: Receipt
- ID-549: Bill Payment Modes
- ID-533: Ledger
- ID-530: Cheque Bounce Entry
- ID-524: Summary Report
- ID-536: Backup
- ID-537: Restore

6.2 Description of stories

- **ID-464: Account Maintenance**

This functionality will be used to create and maintain four types of accounts stated below.

- Customer Account
- Supplier Account
- Bank Account
- Employee Account

User should be able to create, update and delete any type of account. Each account will contain some specific information based on the account type and some details common to all the accounts.

- Account Name:

  This will be unique name assigned to the account.

- Account Number:

Every account will be having a unique account number assigned to it.

- Account Type:

  This is specific to bank account and will contain account type such as saving account, loan account etc.

- Open Balance:

  Every account will be having opening balance of the day. This is the credit in the account at the start of the day.

- Open Balance Date:

  This is the date corresponding to the open balance.

- Address:

  This is the address of the account holder along with city and PIN.

- Contact Details:

  This will contain the contact no of the account holder.

Linked Requirements

- ID-749: Customer Account
- ID-750: Supplier Account
- ID-751: Bank Account
- ID-752: Employee Account

➢ ID-517: Billing Header

Billing Header consists of following fields:

- Customer Name: System should prompt for various available customer names or it could be filled manually.
- City: System should prompt for various available city names or it could be filled manually.
- Through: This field should show all distinct entries till now.
- Date: This will be a valid date in dd/mm/yyyy format.
- Bill Number: Unique number not used till now [default- MaxBillNumber - 1].
- Add/View Customer detail: This redirects to Account Maintenance form to update/add the detail of customer.

Linked Requirements:

- ID-763: Billing Header

➢ ID-518: Billing Body

BILLING BODY ESSENTIALLY CONSISTS OF ITEM ENTRY OF VARIOUS ITEMS TAKEN BY CUSTOMER ,ITS RATE , QUANTITY , DESCRIPTION E.T.C. BILL CONSISTS OF FOLLOWING ENTRIES:

- Item Company - name of the company of the item being sold
- Item Type- type of the item eg, bedsheet, saree
- Item Name - specific names of one type
- Item Detail - text details it may also contain details like no of dozens, meters, quantity.
- Quantity – no of items of number of bundles

- Meter – no of meters if item is of meter type else empty
- Rate – rate of item
- Amount – meter * rate if meter type else quantity * rate
- An item is meter type if it is sold in units of meters (eg 2.5mt of suit cloth) and non meter types are those items which are sold as individual pieces.

Linked Requirements:

- ID-764: Billing Body

- **ID-519: Billing Footer**

  Billing Header consists of following fields:

  - RG Total: Total cost of RG(Return Good) items (Non Editable)
  - Total: Sum of cost of all Non RG items (Non Editable)
  - Expense% / Discount %: Decidable by client [Default: 2].
  - Expenses or Discount: Evaluated from above two fields (Total*(Exp/Dis)/100)
  - Transport Name
  - Transport Number
  - Transport Charge: Charge against the Transport Name and City (in Header).
  - Grand Total: Evaluated from the values in RG Total, Total, Transport Charge, and Expense/Discount.
  - Number of bales: An integer value [default: 1]

- Note

Linked Requirements:

- ID-765: Billing Footer

➢ ID-520: Billing_RG Entry

In some scenario goods sold to customer can be returned due to reasons like Damage , Not Required e.t.c. In such cases whenever new bill of customer is made then those items are inserted and its amount is deducted from total amount,so RG item entry should be same as that of Billing entry but it should hold additional functionality by which wherever item is entered then it will show max 5 result of bill number and item detail which consist of those items.

Linked Requirements:

- ID-764: Billing Body

➢ ID-465: Price List

For maintenance of inventory item price list has to be maintained. Price list consists of various items namely:

- Item Company: This attribute containing the name of the companies to deal with.
- Item: TypeThis attribute is containing the type of items.
- Item Name*: An item type can have many names. So this attribute is differentiating it.
- Item Price*: Price of the items in Indian rupee.

- Item Quantity: Total quantities of the particular item type.

Here first four are used for maintenance of price list while last is used for keeping track of particular item. All these items should be present in printable format where we can have print after item selection. Items one added to database can be modified. Item Name, Item Type, Item Company together ensures that item are unique in nature. Item may also contain entry for meter type for calculation of quantity and meter while billing.

Linked Requirements:

- ID-**759: Price List**

➢ ID-535: Bill Printing

There is a particular format of printing Bill its not actually printed in A4 size paper but a paper which is less in size than A4 its specification are A4 with top margin = 0.25 inch , Left = 0.56 , Bottom = 1.86 , Right 0.56 inches.

While printing bill there are certain things to be remembered: -

- ID-**759: Price List**
- Output should be constrained paper size as it is been set by the user (size is mentioned above).
- It should be intelligent in a way if expenses/discount = 0 then it should not be shown, if transport charge = 0 then it should not be shown, if note field is empty then it should not be shown at the final output.

- In transport name only word ‘transport’ should be written
- Bill print should be sorted with first sorting item type then company then item rate.
- RG should always come at the last of bill and should be indicated in bold letter.
- There should be sum of quantity which is the sum of total quantity excluding RG part.
- One bill can be extended to multiple pages.
- Closing Balance should be indicated in the Bill.
- In Grand Total either “Net Amount To Pay” or “Paid By Cash” should be mentioned indicating Credit or Cash Bill Type.
- Every field indicating rupees should be indicated with Indian rupees sign (as <rupee> 1234.00).
- Every rupee should be rounded off using previous rules.
- Its preferable that grand total doesn’t contain any paisa part.

Linked Requirements:

- ID-**766: Bill operations**

➢ ID-538: Receipt

Once good is sold customer gets bill that contains details of all the items sold. When customer makes payment for the bill, he will be given a receipt containing following details.

- Receipt No:

  This will be a unique number generated automatically for every receipt.

- Date:

  This will contain the date when payment has been made.

- Customer Detail:

  This will contain the name and city of the customer to whom receipt is being issued.

- Bill No:

  Every receipt will be having bill no(may be more than one) against which the payment is done. However this field can be set to blank if payment is done in advance.

- Through:

  This field will contain the name of the person through whom payment is made.

- Amount:

  This will be the billed amount to be paid. User may or may not offer discount on this.

- Cash Discount:

  This will be the discount offered by the customer to the user on the amount.

- Total:

This will be the amount after deducting the discount and that has to be paid by the customer.

- Bank and DD/Check Details;

  If customer is making payment by Cheque/DD, corresponding Cheque/DD no. along with name of the issuing bank will be filled here.

- Note:

  This field can contain some additional comments if user wishes to write some.

Linked Requirements:

- ID-768: Receipt
- ID-749: Customer Account
- ID-750: Supplier Account
- ID-751: Bank Account
- ID-752: Employee Account

➢ ID-549: Bill Payment Modes

There can be two modes:

- Cash Payment: No receipt is made in case of cash payment. In cash payment default payment type is set to discount. However it's in user hand to give discount or take expenses.
- Credit Payment: Payment (Bill) and Receipt of the customer is stored in his Ledger. Default payment type is set to expenses (2%).

Linked Requirements:

- ID-765: Billing Footer

➢ ID-533: Ledger

Ledger is a page which contains all information of transaction by a particular customer. Each customer will have its own ledger. It consists of following fields:

- Customer name
- City
- Print Date
- Page Number
- Open Balance
- Date of transaction
- Transaction Type – Bank , cash , check , Bill , Receipt
- Transaction Detail – Bill Number , Receipt Number , Check Number , Check Bounce Id
- Manual Bill / Receipt Number
- Expenses
- Payment
- Receipt
- Rate of interest (not to be shown at printout)
- Interest Amount (not to be shown at printout)
- Payment total

- Receipt total

Linked Requirements:

- ID-770: Ledger

➢ ID-530: Cheque Bounce Entry

It will contain the following entries:

- ID-770: Ledger
- Receipt Number: Containing the unique number of the cheque receipt.
- Date: Indian format date(dd/mm/yyyy) mentioned on the cheque.
- Bounce Charge: Predefined charge that will apply on cheque bounce.
- Bank Name: Name of the bank mentioned on the cheque.
- Customer Name: This will contain the name of the customer who has given this cheque.
- City: This entry is for the resident city of customer.
- Amount: Amount in Indian rupee mentioned on cheque.
- Note: If require any note then user can use this field.

On entering cheque number all entries should automatically be filled but entries can change by user (cheque number is present because of receipt entry or bank entry). On saving bounce entry it should be reflected in the party ledger in the payment side as this is the amount to be taken from the customer. Actually this will tend to nullify the effect of receipt. On entering cheque details if its details are already available in the database then should be auto filled (but all fields can be modifiable).

Linked Requirements:

- ID- 776: Cheque Bounce Entry

➢ ID-524: Summary Report

This functionality will be used in getting details of all the transactions done between specific dates. User can select any period and get the details of all the transactions done in that period. It should contain following entries.

- Date From

  This is the starting date of the period to be selected.

- Date To

  This is the ending date of the period to be selected.

- Date

  This will be the date when the summary report has been generated(current date).

- Customer Name

  Name of the customer with whom business is done.

- Details

  It will contain the details of the transaction.

- Expenses

  It will be having details of extra expenses done with the transaction.

- Payment

  This field will contain the payment details of the transaction.

- Receipt

This will be having details of receipt that has been issued for that transaction.

- Total

  This will be the total of amounts of all the transactions.

Linked Requirements:

- ID-771: Summary
- ID-750: Summary Report

- **ID-536: Backup**

  An easy interface should be provided to the user to take back up of all the data for safety. He can choose options for automated, manual or overwrite backup.

  Also following functions should be given in the interface.

  - User can change the location where the backup is to be stored.
  - When user wishes to take the backup the default location will be the location chosen for the previous backup made.
  - User can take backup on daily, weekly or monthly basis.

- **ID-537: Restore**

An easy interface should be provided for restoring the backups. This functionality should be password protected so that only authentic person can restore the backup. Interface for restoring the backups will be having following features.

- It will ask for the location of the backup file from where this to be restored.
- It will authenticate the user before restoring the backup file.
- After authentication and selection of file it should prompt for the further confirmation to restore the backup file.

Chapter 7

Requirements

7.1 Description of requirements

- **ID-749: Customer Account**

  An interface should be provided to the user to create and maintain customer accounts. Also proper check for data types and constraints as given in the table below should be implemented.

Customer Account field			
Account Name*	string	100	Valid and unique account name
City*	string	50	Valid city name (should not contain numbers or special characters, dot is allowed).
Open Balance	numeric	12,2	It should be set to 0.00 if left blank , should be appended with decimal and zero if not already done
Open Balance Date	Date		default value start of finincial year , dd/MM/yyyy
Address	string	500	Multi Line Field
Pincode	String	6	Valid 6 digit PIN code
Phone Number	Number	10	Valid 10 digit phone no(with STD code)
Mobile Number	Number	10	Valid 10 digit mobile no.
Note	Text	Infinite	Multi Line field

Few key points to remember in this interface.

- Account name is mandatory field and should be present before Account creation, updation
- Account name + City is a unique key
- Open balance should automatically be set to 0 when left blank
- Open balance date should automatically be set to begin of financial year
- Open Balance Date should be in the format dd/MM/yyyy
- Address is a multiline field
- Note is a multiline field
- Open Balance should automatically be appended to .00 if decimal is not already present

Implementation Details

- Implemented By: Siddharth Jain
- Source Code: Account Maintenace.cs
- Database Tables/Views: view_account_customer, view_account_bank, view_account_supplier,view_account_employee

Validation Details

- Validated By: Shabbir Hussain

Test Cases:

- ID-63: Test case for customer Account Maintenance form "Save " Button
- ID-62: Test case for customer Account Maintenance form "Save Print" Button

- **ID-768: Receipt**

This functionality requires an easy interface to be provided to the customer for printing and saving the receipts. The interface should also check for the correct type of the data and the constraints given for the every field.

Receipt Fields			
Receipt Number**	Int		Should follow the concept of Bill Number and it should also be open as bill number and some interface should be provided to get information about missing Receipt Number as Bill Number
Customer Name*	String	100	It should have intelligence as in case of bill it should also have option to ass account as in Bill. This also can be either customer name or employee name as in Billing Employee dosenot contain City and has to be taken care off.
City*	String	50	
Receipt Date*	Date		dd/MM/yyyy
Bill Number	String or int (To be decided Later)		in Some cases this field can be empty , in some cases this field may contain number , in some cases this field may contain multiple numbers separated by comma our task is to show bill header and footer information of all bill number specified here
Through	String	200	Intelligence as in bill
Amount*	Numeric	12,2	paisa field , right indent
Amount in words			This is an auto generated field which takes amount and shows it in words like 1234.50 = One Thousand Two Hundred Thirty Four Rupees and Fifty Paisa Only...
Cash Discount	Numeric	12,2	

Total*	Numeric	12,2	Amount + Cash Discount (readonly)
Manual Receipt Number	Int		
Bank Name	String	100	this and below field are rarely used field so a check box is to be given so that whenever user this checkbox these fields will get visible and user will be allowed to enter information
Check Number/DD Number	String	50	
Note	Text		Multiline

Implementation Details

- Implemented By: Rishi Garg
- Source Code: Receipt.cs, Receipt.Designer.cs
- Database Tables/Views: view_receipt

Validation Details

- Validated By: Shabbir Hussain

Test Cases:

- ID-72: For checking the link of Receipt Form
- ID-73: For validating the format of Receipt form
- ID-74: For validating the save button on receipt form
- ID-75: For validating the Add button on receipt form
- ID-76: For validating the print button on receipt form
- ID-77: For validating the delete button on receipt form

- **ID-517: Billing Header**

Implementation Detail

- Implemented By: Saurav Kant Jha
- Source Code: Billing.cs
- DataBase: view_customer_bill

Validation Detail

- Validated By: Shabbir Hussain

Test Cases:

- Validated By: Shabbir Hussain
- ID-36: Validation for existing customer
- ID-37: Validation for new customer
- ID-38: Validation for "Through", "Date", "Bill No" Field

- **ID-765: Billing Footer**

Implementation Detail:

- Implemented By: Saurav Kant Jha
- Source Address: Billing.cs
- Database Used: view_customer_bill

Validation Detail:

- Validated By: Shabbir Hussain

Test Cases:

- ID-41: Validation of “save” button with new customer
- ID-42: Validation of “ Clear” button
- ID-43: Validation of “ Save Print” button for existing customer
- ID-44: Validation of “ Print All” button for existing customer
- ID-45: Validation of “ Save Print” button for new customer
- ID-46: Validation of “ Print All” button for new customer
- ID-47: Validation of “ Check Item” button

➢ ID-750: Supplier Account

Supplier account will hold all the details about the supplier that are given in the story. Each filed will be checked for the correct data type and constraints mentioned below.

Supplier Account Fields			
Account Name*	String	100	Valid and unique account name
City*	String	50	Valid city name (should not contain numbers or special characters).
State*	String	50	Valid city name (should not contain numbers or special characters, dot is allowed).
Open Balance	Numeric	12,2	It should be set to 0.00 if left blank , should be appended with decimal and zero if not already done
Open Balance Date	Date		default value start of finincial year ,

			dd/MM/yyyy
Address	String	500	Multi Line Field
Pincode	number	6	Valid 6 digit PIN code
Phone Number	Number	10	Valid 10 digit phone no(with STD code)
Mobile Number	Number	10	Valid 10 digit mobile no.
Note	Text	Infinite	Multi Line field

Account Name + City + State combination should be unique

Implementation Details

- Implemented By: Siddharth Jain
- Source Code: Account Maintenace.cs
- Database Tables/Views: view_account_supplier

Validation Details

- Validated By: Shabbir Hussain

Test Cases:

- ID-64: Test case for supplier Account Maintenance form "Save " Button
- ID-65: Test case for supplier Account Maintenance form "Save Print" Button

➢ ID-751: Bank Account

This functionality will help user to maintain details about various bank accounts. This interface should also check for correct data type in the fields and also check the constraints mentioned in the table below.

Bank Account Field			
Account Name	String	100	Valid and unique account name
Account Number	String	50	Valid account number
Account Type	String	Bit	only two value possible -> either Current or Loan Account
Open Balance	Numeric	12,2	setted to appripriate decimal value
Open Balance Date	Date		dd/MM/yyyy defaults to finincial year start date
Note	Text		Any comment if user wishes to make.

Implementation Details

- Implemented By: Siddharth Jain
- Source Code: Account Maintenace.cs
- Database Tables/Views: view_account_bank

Validation Details

- Validated By: Shabbir Hussain

Test Cases:

- ID-67: Test case for bank Account Maintenance form "Save " Button
- ID-66: Test case for bank Account Maintenance form "Save Print" Button

- **ID-752: Employee Account**

Employee account will be having all the details about employee working in the shop. An easy interface must be provided to the user to create, modify and delete the account. Also correct data type and constraint's check should be done properly as given in the table below.

Employee account Fields			
Account Name*	String	100	Valid and unique account name
Open Balance	numeric	12,2	It should be set to 0.00 if left blank , should be appended with decimal and zero if not already done
Open Balance Date	Date		default value start of finincial year , dd/MM/yyyy
Address	String	500	Multi Line Field
Pincode	number	6	Valid 6 digit PIN code
Mobile Number	number	10	Valid 10 digit mobile no
Note	Text		Any additional comment that user wishes to make.

Implementation Details

- Implemented By: Siddharth Jain
- Source Code: Account Maintenace.cs
- Database Tables/Views: view_account_employee

Validation Details

- Validated By: Shabbir Hussain

Test Cases:

- ID-68: Test case for customer Account Maintenance form "Save " Button
- ID-69: Test case for customer Account Maintenance form "Save Print" Button

- **ID-764: Billing Body**

Item Company	String	50	Valid name of the company
Item Type	String	50	Item type will be of format defined in the story
Item Name	String	50	Item name e.g. shirts,T-shirt etc.
Item Detail	String	100	
Quantity	Int		
Meter	Float		
Rate	Numeric	12,2	All rate and amount and every where whereever rs is there it is mandetory that it is right indented
Amount	Numeric	12,2	

Internal Fields			
Item	int		but it should not be linked using foreign key with item as it can be

Code (refrence to item)			case that item do not exist but is present in bill because of conditions like item is deleted , entry only consists of item code e.t.c.
is rg	bit		weather this perticular item belongs to rg?
Checked	bit		used to tell weather this item was chwecked while sending bill

- if item company and or item type is not filled by user then first it should check in the database if entry pertaining to it exists if yes and they are not empty then fill them .if no then copy it from above column
- None of item is necessary bill should be allowed to save even it some item are left to be filled later on.
- But user should be shown error signal and error line where error has been occured before saving.
- every rate , amount should be appended with paisa field if not done already
- paisa should automatically be truncated when there are more than 2 digits after decimal
- item detail may contain various equation using ***** separated with comma and there may also be some text we have to identify equation , add up answer of all equation and if item is of type non meter then put all those in quantity otherwise in meter if unknown then put it into quantity by default
- whenever item information is filled if we found that item rate is available in the database then insert its rate in rate filled and if quantity is also known to us then calculate amount as meter*rate if meter is available otherwise qty * rate and append all rate and amount with paisa field (zeros)

- if amount >100 then round off amount to its nearest integer otherwise it will have 3 division as 0 , 0.5 , 1 and we have to round it off to nearest of three
- item name can be appended at last with item code which is made of adding 100 , remove decimal , append 5 on both sides if code is given then we should calculate rate from this information and display it in rate field
- An option should be given to enter RG details (essentially a button) RG details contains all fields as above and all operations as above with one additional feature
    - Whenever user enters an rg item then our program should search for all (at max recent 5 bills) in which user has taken those item (it should not give any error if item is not present) and if present then on double click item should be automatically be filled
    - This should repeat for each item entered.

Implementation details

- Implemented by: Preetham V
- Source Code : Billing.cs, Billbody.cs

Validation details

- Validated by Shabbir Hussain

Test cases

- **48: BILLING BODY VERIFICATION-PURCHASED ITEMS EXIST IN DATABASE**
- **49: BILLING BODY VERIFICATION-PURCHASED ITEMS INCLUDE SOME NEW ITEMS THAT ARE NOT IN DATABASE**

- 50: Billing Body Verification - RG Data Grid

➢ ID-776: Cheque Bounce Entry

Whenever a check bounces then a cheque bounce entry has to be made which is reflected in the payment side of the customer ledger along with cheque bounce entry one more entry has to be made which is known as cheque bounce penalty and is reflected in the ledger differently. It will also be reflected in the bank ledger as previously we have Credited bank with this amount but now we have to again Debit the bank. And penalty amount will go in account. And it will also be reflected in the customer account in the payment side indicating we have to take some extra amount.

It contains following field

Voucher Number	Int		This receipt number, it has no relation with the receipt given while making payment can also be said as Cheque bounce entry SNO.
Date	Date		Defaults to today's date
Cheque Number	String	100	this may correspond to one of the cheque entered earlier in the Receipt whenever we enter a cheque number it that number already exist in the database then it should display all information of that cheque (no action has to be taken if cheque number dose not exists but even then also entry is allowed to be saved.
Bounce charge	Numeric	12,2	
Bank Name	String	100	

Customer Name	String	100	
City	String	50	
Amount	Numeric	12,2	
Note	Text		

- **766: Bill Operations**

Only thing which distinguish two bill is the bill number itself all other quantity can repeat like there can be bill of same customer with different bill name person can take same items in another bills so such dependency should not be think and only thing which has to be taken care of is bill number.

Before saving bill there are certain things which has to be checked which are as follows if user is saving the bill (not updating) then bill number should not already exist if bill number is not the max bill number + 1 this implies that bill number reference to some missing bill number so date has to be checked weather current bill date follows that it lies in between min and max allowable dates.

- if there is some mistake in item as
    - item company + item type + item name + item code = either empty of white spaces
    - any quantity is empty
    - any rate is not filled
    - any amount id not present
- in such scenarios user should be prompted that there are some error in the bill a red has to be shown and line in which there are error has to be colored with red but user should not be stopped from saving or printing just we have to give error message its choice of user at that point of time what to do.
- Internally in the front end itself we can create some unique id field to map item in check field and item in Main Bill

Bill Save

Bill should be allowed to save even if conditions above indicate fail based on user choice. While Saving bill if particular item is not present in the database then it has to be inserted in the database , if rate of certain item has been changed then an option has to be provided to update all rates to update all rates stored in the database which should by default not update

Bill Update

for updating bill first user has to choose which bill has to be updated. Then whole bill should load in out screen and user is allowed to change whatever he wants to change in the bill once changes has been done when user updates bill latest copy of the bill should be present in the database.

Bill Delete

Bill is allowed to be deleted whenever user deletes a bill then whole is created in the bill number that has to be maintained appropriately. An interface has to be provided to know which all bill number is left blank in between

Bill Print

Whenever user wants to print a bill then a particular format is specified which is necessary to keep one can found that in the resources here which will be uploaded soon

- Bill print has to maintain some sort of intelligence as whenever something in footer is empty (which can be transport , discount/expenses , discount% , expense % , note) or zero then it should not appear in the bill
- RG should always come at the end of bill and should be marked bold
- there should be quantity sum which is different for both RG and NON RG
- Bill output for printing should be in sorted order i.e it should be first sorted according to item type then according to Item Company then according to item price.
- Printing Paper for Bill Print is not of A4 size so care has to be taken while creating output paper size.

Bill Checking

Whenever user completes bill formation then he checks the bill in a way that every item written in the bill is correct or not

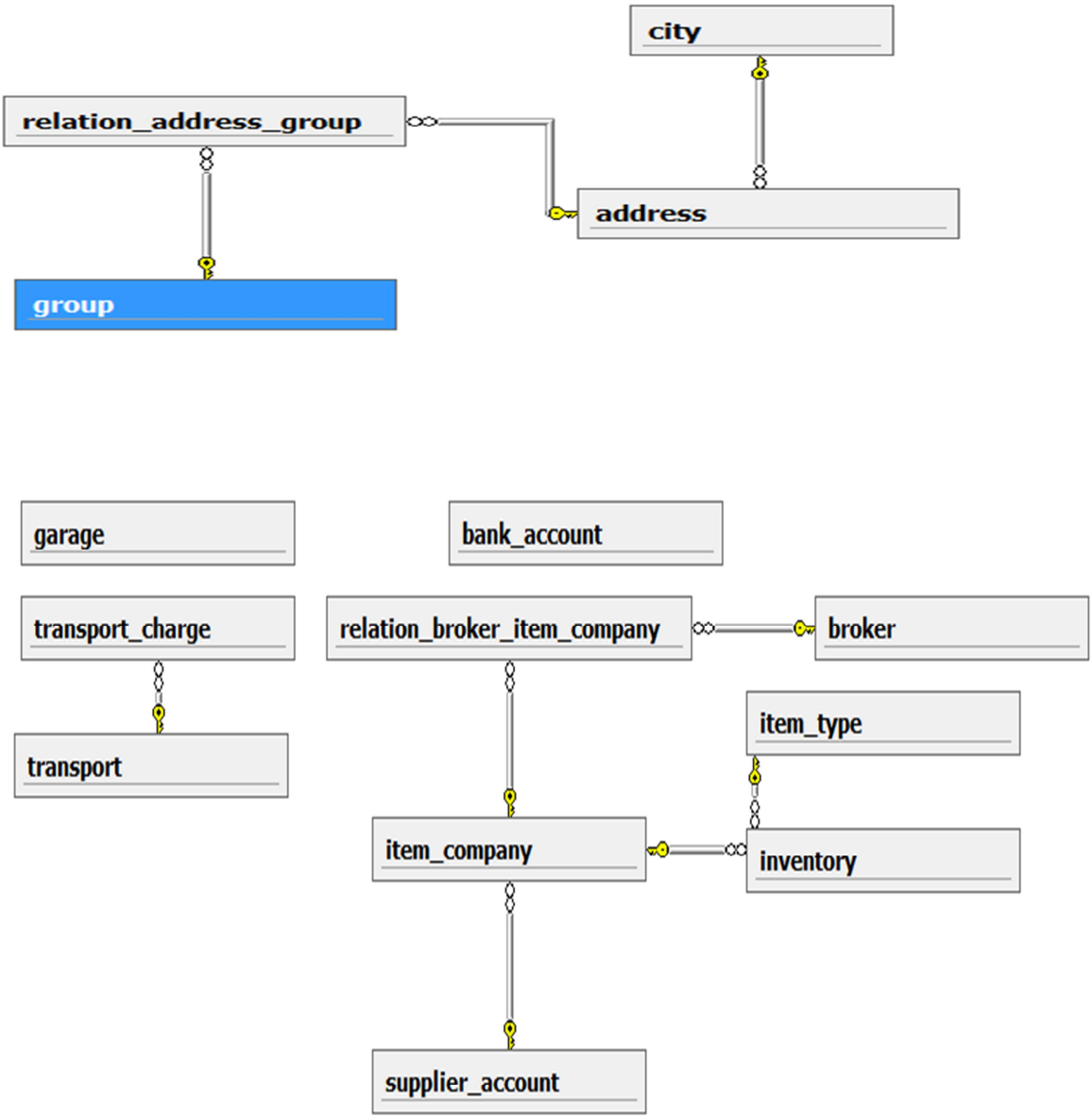
Fields in Checking	
Button	enables disabled checkbox
Checkbox	gets disabled when once checked
Error Check Box	indicates that there is some error in this particular item and it should be reflected in the billing form with RED marked and setting Error signal and its color
Item Company	Valid Name of the company
Item Type	
Item Name	Name of item e.g. shirts, T-shirts etc.
Item Detail	
Item Quantity	
Item Rate	

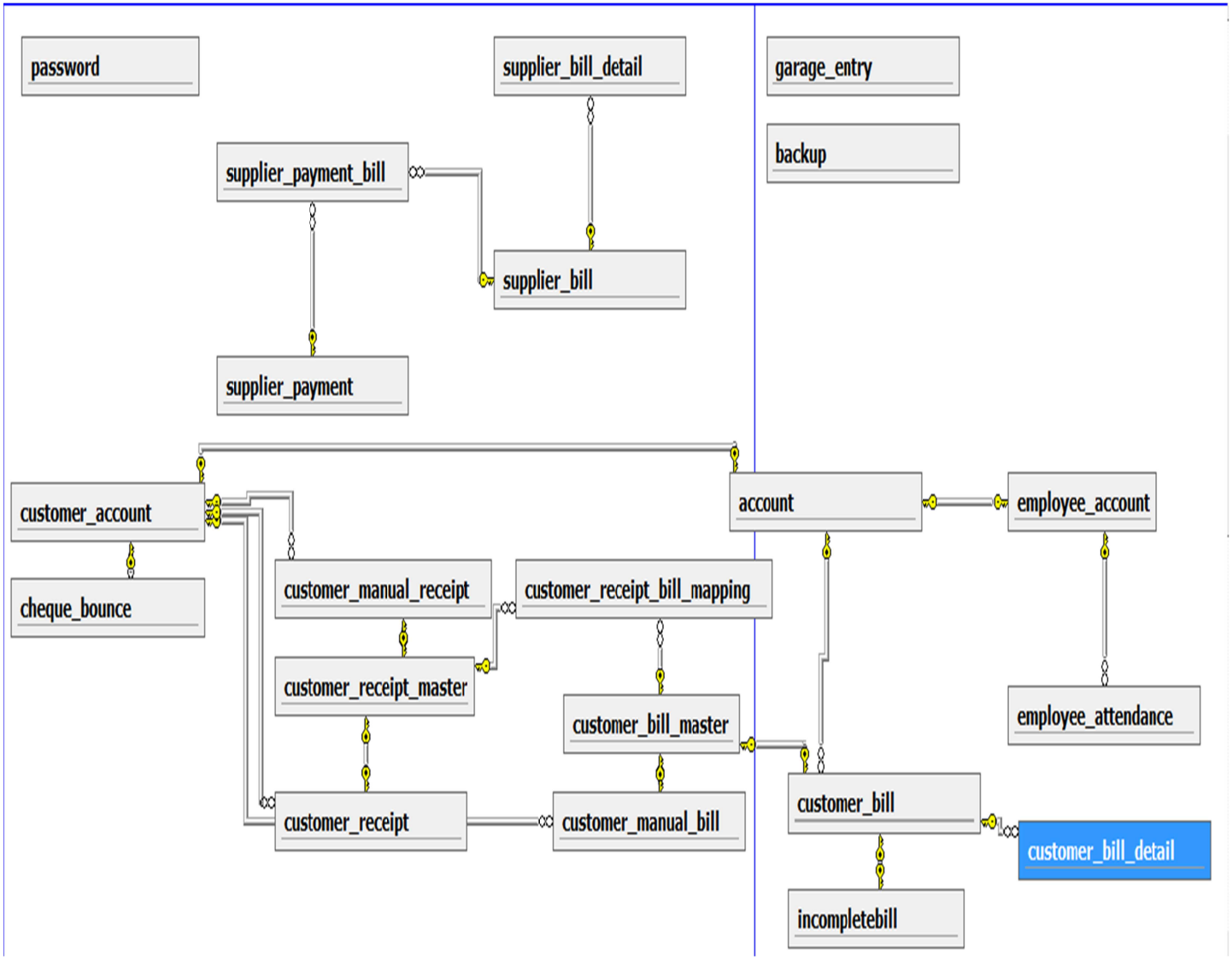
Print All: It will show all bill Number serial wise in a printable format

Chapter 8

Database Description

8.1 E-R Diagrams




password
supplier_bill_detail
garage_entry
backup
supplier_payment_bill
supplier_bill
supplier_payment
customer_account
account
employee_account
cheque_bounce
customer_manual_receipt
customer_receipt_bill_mapping
customer_receipt_master
customer_bill_master
employee_attendance
customer_receipt
customer_manual_bill
customer_bill
customer_bill_detail
incompletebill

8.2 Description of Views

Database: db_address

View Name: View_address

Details:

S. No.	Column	Alias	Table
1	address_id		address
2	address_name		address
3	city_name		city
4	city_state		city
5	address_pincode		address
6	address_phone_number		address
7	address_mobile_number		address

Database: db_common

View Name: View_company_broker

Details:

S. No.	Column	Alias	Table
1	item_company_name	company_name	item_company
2	broker_name		broker
3	relation_id		relation_broker_item_company

View Name: View_garage

Details:

S. No.	Column	Alias	Table
1	garage_name		garage
2	address_name	garage_address	address (db_address)
3	city_name	garage_city	city (db_address)
4	city_state	garage_state	city (db_address)

View Name: View_inventory

Details:

S. No.	Column	Alias	Table
1	item_id		inventory
2	item_company_name	item_company	item_company
3	item_type_name	item_type	item_type
4	item_name		inventory
5	item_code		inventory
6	item_is_meter_type		item_type
7	item_rate		inventory
8	item_quantity		inventory

View Name: View_supplier

Details:

S. No.	Column	Alias	Table
1	supplier_account_id	supplier_id	supplier_account
2	supplier_name		supplier_account
3	city_name	supplier_city	city (db_address)

4	city_state	supplier_state	city (db_address)
5	address_name	supplier_address	address (db_address)
6	address_pincode	supplier_pincode	address (db_address)
7	address_phone_number	supplier_phone_number	address (db_address)
8	address_mobile_number	supplier_mobile_number	address (db_address)
9	open_balance		supplier_account
10	open_balance_date		supplier_account
11	phone_number		supplier_account
12	mobile_number		supplier_account
13	note	supplier_note	supplier_account

View Name: View_transport

Details:

S. No.	Column	Alias	Table
1	transport_id		transport
2	transport_name		transport
3	transport_city		transport_charge
4	transport_rate		transport_charge

Database: db_VardhamanTextile

View Name: View_account_bank

Details:

S. No.	Column	Alias	Table
1	bank_id		bank_account (db_common)
2	bank_name		bank_account (db_common)
3	bank_account_number		bank_account (db_common)
4	bank_account_type		bank_account (db_common)
5	bank_open_balance		bank_account (db_common)
6	bank_open_balance_date		bank_account (db_common)
7	bank_note		bank_account (db_common)

View Name: View_account_customer

Details:

S. No.	Column	Alias	Table
1	account_id	customer_id	account
2	customer_name		customer_account
3	customer_city		customer_account
4	account_open_balance	customer_open_balance	account
5	account_open_balance_date	customer_open_balance_date	account
6	account_note	customer_note	account
7	address_id		View_address (db_address)
8	address_name	customer_address	View_address (db_address)
9	city_state	customer_state	View_address (db_address)

10	address_pincode	customer_pincode	View_address (db_address)
11	account_phone_number	customer_phone_number	account
12	account_mobile_number	customer_mobile_number	account

View Name: View_account_customer_total

Details:

S. No.	Column	Alias	Table
1	customer_id		customer_account
2	customer_name		customer_account
3	customer_city		customer_account
4	account_open_balance	customer_ open_balance	account
5	account_open_balance_date	customer_open_balance_date	account
6	address_name	customer_address	address (db_address)
7	city_state	customer_state	city (db_address)
8	address_pincode	customer_pincode	address (db_address)
9	address_phone_number	customer_phone_number	address (db_address)
10	address_mobile_number	customer_mobile_number	address (db_address)
11	account_note	customer_note	account

View Name: View_account_employee

Details:

S. No.	Column	Alias	Table
1	employee_id		employee_account
2	employee_name		employee_account
3	employee_payment		employee_account

4	account_open_balance	employee_open_balance	account
5	account_open_balance_date	employee_open_balance_date	account
6	account_note	employee_note	account
7	address_id		account
8	address_name	employee_address	address (db_address)
9	city_name	employee_city	address (db_address)
10	city_state	employee_state	address (db_address)
11	address_pincode	employee_pincode	address (db_address)
12	account_phone_number	employee_phone_number	account
13	account_mobile_number	employee_mobile_number	account

View Name: View_account_employee_total

Details:

S. No.	**Column**	**Alias**	**Table**
1	employee_id		employee_account
2	employee_name		employee_account
3	account_open_balance	employee_open_balance	account
4	account_open_balance_date	employee_open_balance_date	account
5	address_name	employee_address	address (db_address)
6	city_name	employee_city	city (db_city)
7	city_state	employee_state	city (db_city)
8	address_pincode	employee_pincode	address (db_address)
9	address_phone_number	employee_phone_number	address (db_address)
10	address_mobile_number	employee_mobile_number	address (db_address)

View Name: View_account_supplier

Details:

S. No.	Column	Alias	Table
1	supplier_id		View_supplier (db_common)
2	supplier_name		View_supplier (db_common)
3	supplier_city		View_supplier (db_common)
4	supplier_state		View_supplier (db_common)
5	supplier_address		View_supplier (db_common)
6	supplier_pincode		View_supplier (db_common)
7	supplier_phone_number		View_supplier (db_common)
8	supplier_mobile_number		View_supplier (db_common)
9	open_balance		View_supplier (db_common)
10	open_balance_date		View_supplier (db_common)
11	phone_number		View_supplier (db_common)
12	mobile_number		View_supplier (db_common)
13	supplier_note		View_supplier (db_common)

View Name: View_address

Details:

S. No.	Column	Alias	Table
1	address_id		View_address (db_address)
2	address_name	address	View_address (db_address)
3	city_name	city	View_address (db_address)
4	city_state	state	View_address (db_address)
5	address_pincode	pincode	View_address (db_address)
6	address_phone_number	phone_number	View_address (db_address)
7	address_mobile_number	mobile_number	View_address (db_address)

View Name: View_bill_detail

Details:

S. No.	Column	Alias	Table
1	customer_name		View_sale_account, customer_bill, View_customer_manual_bill
2	customer_city		View_sale_account, customer_bill, View_customer_manual_bill
3	cbill_number		View_sale_account, customer_bill, View_customer_manual_bill
4	cbill_date		View_sale_account, customer_bill, View_customer_manual_bill
5	cbill_payment_mode		View_sale_account, customer_bill, View_customer_manual_bill
6	cbill_rg_total		View_sale_account, customer_bill, View_customer_manual_bill
7	cbill_exp_dis_per		View_sale_account, customer_bill, View_customer_manual_bill
8	cbill_transport_id		View_sale_account, customer_bill,

			View_customer_manual_bill
9	cbill_transport_charge		View_sale_account, customer_bill, View_customer_manual_bill
10	cbill_isdiscount		View_sale_account, customer_bill, View_customer_manual_bill

View Name: View_cheque_bounce

Details:

S. No.	Column	Alias	Table
1	customer_id	cb_customer_id	customer_account
2	customer_name	cb_customer_name	customer_account
3	customer_city	cb_customer_city	customer_account
4	cheque_bounce_voucher_ number	cb_voucher_number	cheque_bounce
5	cheque_bounce_date	cb_date	cheque_bounce
6	cheque_number	cb_cheque_number	cheque_bounce
7	cheque_bounce_charge	cb_charge	cheque_bounce
8	bank_name	cb_bank	bank_account (db_common)
9	cheque_bounce_amount	cb_amount	cheque_bounce
10	cheque_bounce_note	cb_note	cheque_bounce

View Name: View_city

Details:

S. No.	Column	Alias	Table
1	city_id		city (db_address)
2	city_name	city	city (db_address)

3	city_state	state	city (db_address)

View Name: View_customer_bill

Details:

S. No.	Column	Alias	Table
1	customer_id	cbill_customer_id	customer_account
2	customer_name	cbill_customer_name	customer_account
3	customer_city	cbill_customer_city	customer_account
4	cbill_number		customer_bill
5	cbill_through		customer_bill
6	cbill_date		customer_bill
7	cbill_update_date		customer_bill
8	cbill_payment_mode		customer_bill
9	cbill_rg_total		customer_bill
10	cbill_total		customer_bill
11	cbill_exp_dis_per		customer_bill
12	cbill_transport_id		customer_bill
13	cbill_transport_charge		customer_bill
14	cbill_number_of_bales		customer_bill
15	cbill_note		customer_bill
16	cbill_isdiscount		customer_bill

View Name: View_customer_bill_detail

Details:

S. No.	Column	Alias	Table
1	cbill_detail_id		customer_bill_detail
2	cbill_id		customer_bill

3	cbill_number		customer_bill
4	item_id		customer_bill_detail
5	item_company		customer_bill_detail
6	item_type		customer_bill_detail
7	item_name		customer_bill_detail
8	item_code		customer_bill_detail
9	item_detail		customer_bill_detail
10	item_quantity		customer_bill_detail
11	item_meter		customer_bill_detail
12	item_rate		customer_bill_detail
13	is_returned_good		customer_bill_detail
14	ischecked		customer_bill_detail

View Name: View_cutomer_manual_bill

Details:

S. No.	**Column**	**Alias**	**Table**
1	customer_id	cmbill_customer_id	customer_account
2	customer_name	cmbill_customer_name	customer_account
3	customer_city	cmbill_customer_city	customer_account
4	cmbill_id		customer_manual_bill
5	cmbill_number		customer_manual_bill
6	cmbill_date		customer_manual_bill
7	cmbill_update_date		customer_manual_bill
8	cmbill_payment_mode		customer_manual_bill
9	cmbill_total		customer_manual_bill
10	cmbill_expenses_discount_per		customer_manual_bill
11	cmbill_transport_charge		customer_manual_bill
12	cmbill_note		customer_manual_bill

13	cmbill_isdiscount		customer_manual_bill

View Name: View_customer_manual_receipt

Details:

S. No.	Column	Alias	Table
1	cmreceipt_customer_id		customer_manual_receipt
2	customer_name	cmreceipt_customer_name	customer_account
3	customer_city	cmreceipt_customer_city	customer_account
4	cmreceipt_id		customer_manual_receipt
5	cmreceipt_number		customer_manual_receipt
6	cmreceipt_date		customer_manual_receipt
7	cmreceipt_bill_number		customer_manual_receipt
8	cmreceipt_through		customer_manual_receipt
9	cmreceipt_amount		customer_manual_receipt
10	cmreceipt_cash_discount		customer_manual_receipt
11	cmreceipt_total		customer_manual_receipt
12	bank_name		bank_account (db_common)
13	cmreceipt_cheque_dd_number		customer_manual_receipt
14	cmreceipt_note		customer_manual_receipt

View Name: View_customer_receipt

Details:

S. No.	Column	Alias	Table
1	creceipt_customer_id		customer_receipt
2	customer_name	creceipt_customer_name	customer_account

3	customer_city	creceipt_customer_city	customer_account
4	creceipt_id		customer_receipt
5	creceipt_number		customer_receipt
6	creceipt_date		customer_receipt
7	creceipt_bill_number		customer_receipt
8	creceipt_through		customer_receipt
9	creceipt_amount		customer_receipt
10	creceipt_cash_discount		customer_receipt
11	creceipt_total		customer_receipt
12	bank_name	creceipt_bank	bank_account (db_common)
13	creceipt_cheque_dd_number		customer_receipt
14	creceipt_note		customer_receipt

View Name: View_employee_attendence

Details:

S. No.	Column	Alias	Table
1	employee_id		employee_account
2	employee_name		employee_account
3	attendance_id		employee_attendence
4	attendance_date		employee_attendence
5	attendance		employee_attendence
6	amount		employee_attendence
7	attendecne_note		employee_attendence

View Name: View_garage_entry

Details:

S. No.	Column	Alias	Table
1	garage_entry_id		garage_entry
2	garage_name		garage (db_common)
3	supplier_name		supplier_account (db_common)
4	garage_lr_number		garage_entry
5	garage_date		garage_entry
6	garage_private_marka_number		garage_entry
7	garage_number_of_bales		garage_entry
8	garage_slip_number		garage_entry
9	garage_note		garage_entry

View Name: View_inventory

Details:

S. No.	Column	Alias	Table
1	item_id		view_inventory (db_common)
2	item_company		view_inventory (db_common)
3	item_type		view_inventory (db_common)
4	item_name		view_inventory (db_common)
5	item_code		view_inventory (db_common)
6	item_is_meter_type		view_inventory (db_common)
7	item_rate		view_inventory (db_common)
8	item_quantity		view_inventory (db_common)

View Name: View_sale_account

Details:

S. No.	Column	Alias	Table
1	customer_id		View_account_customer, View_account_employee
2	customer_name		View_account_customer, View_account_employee
3	customer_city		View_account_customer, View_account_employee
4	customer_open_balance		View_account_customer, View_account_employee
5	customer_open_balance_date		View_account_customer, View_account_employee

View Name: View_supplier_bill

Details:

S. No.	Column	Alias	Table
1	sbill_id		supplier_bill
2	supplier_name		supplier_account (db_common)
3	city_name	city	city (db_address)
4	city_state	state	city (db_address)
5	broker_name	broker	broker (db_common)
6	garage_name	garage	garage (db_common)
7	sbill_number		supplier_bill
8	sbill_date		supplier_bill
9	sbill_lr_number		supplier_bill
10	sbill_total		supplier_bill
11	sbill_note		supplier_bill

View Name: View_supplier_payment

Details:

S. No.	Column	Alias	Table
1	spayment_id		supplier_payment
2	spayment_date		supplier_payment
3	supplier_name		supplier_account (db_common)
4	city_name	city	city (db_address)
5	city_state	state	city (db_address)
6	spayment_bill_number		supplier_payment
7	spaymant_total		supplier_payment
8	bank_name	bank	bank_account (db_common)
9	spayment_discount		supplier_payment
10	broker_name	broker	broker (db_common)
11	spayment_cheque_dd_number		supplier_payment
12	spayment_cheque_dd_date		supplier_payment
13	spayment_cheque_dd_duedate		supplier_payment
14	spayment_grand_total		supplier_payment
15	spayment_amount		supplier_payment
16	spayment_note		supplier_payment

View Name: View_transport

Details:

S. No.	Column	Alias	Table
1	transport_id		View_transport (db_common)
2	transport_name		View_transport (db_common)
3	transport_city		View_transport (db_common)

4	transport_rate		View_transport (db_common)

8.3 Description of stored procedures

Database: db_address

Stored Procedure: delete_address

Description: This procedure deletes a row from the address table.

Stored Procedure: delete_city

Description: This procedure deletes a row from the city table.

Stored Procedure: delete_group

Description: This procedure deletes a row from the group table.

Stored Procedure: delete_relation_address_group

Description: This procedure deletes a row from the relation_address_group table.

Stored Procedure: insert_address

Description: This procedure is used to insert a row into the address table.

Stored Procedure: insert_city

Description: This procedure is used to insert a row into the city table.

Stored Procedure: insert_group

Description: This procedure is used to insert a row into the groupt table.

Stored Procedure: insert_relation_address_group

Description: This procedure is used to insert a row into the relation_address_group table.

Stored Procedure: update_address

Description: This procedure is used to update the values of a row in the address table.

Stored Procedure: update_city

Description: This procedure is used to update the values of a row in the city table.

Stored Procedure: update_group

Description: This procedure is used to update the values of a row in the group table.

Stored Procedure: update_relation_address_group

Description: This procedure is used to update the values of a row in the relation_address_group table.

Database: db_common

Stored Procedure: delete_bank_account

Description: This procedure deletes a row from the bank_account table.

Stored Procedure: delete_broker

Description: This procedure deletes a row from the broker table.

Stored Procedure: delete_garage

Description: This procedure deletes a row from the garage table.

Stored Procedure: delete_item_type

Description: This procedure deletes a row from the item_type table.

Stored Procedure: delete_supplier

Description: This procedure deletes a row from the supplier_account table.

Stored Procedure: delete_transport

Description: This procedure deletes a row from the transport table.

Stored Procedure: delete_transport_change

Description: This procedure deletes a row from the transport_charge table.

Stored Procedure: insert_bank_account

Description: This procedure is used to insert a row into the bank_account table.

Stored Procedure: insert_broker

Description: This procedure is used to insert a row into the broker table.

Stored Procedure: insert_garage

Description: This procedure is used to insert a row into the garage table.

Stored Procedure: insert_item_type

Description: This procedure is used to insert a row into the item_type table.

Stored Procedure: insert_supplier

Description: This procedure is used to insert a row into the supplier_account table.

Stored Procedure: insert_transport

Description: This procedure is used to insert a row into the transport table.

Stored Procedure: update_bank_account

Description: This procedure is used to update the values of a row in the bank_account table.

Stored Procedure: update_broker

Description: This procedure is used to update the values of a row in the broker table.

Stored Procedure: update_garage

Description: This procedure is used to update the values of a row in the garage table.

Stored Procedure: update_item_type

Description: This procedure is used to update the values of a row in the item_type table.

Stored Procedure: update_supplier

Description: This procedure is used to update the values of a row in the supplier_account table.

Stored Procedure: update_transport

Description: This procedure is used to update the values of a row in the transport table.

Database: db_Vardhman Textile

Stored Procedure: check_bill_number

Description: This procedure is used to check the validity of a customer bill number in the customer_bill table.

Stored Procedure: check_cbill_number

Description: This procedure is used to check the validity of a customer bill number in the customer_bill table providing additional information regarding the position of a provided customer bill number.

Stored Procedure: check_password

Description: This procedure is used to verify the password entered by the user from the password table.

Stored Procedure: check_receipt_number

Description: This procedure is used to check the validity of a receipt number in the customer_receipt table.

Stored Procedure: customer_bill_blank_insertion

Description: This procedure is used to insert a blank (null valued entry) entry into the customer_bill table.

Stored Procedure: delete_account

Description: This procedure is used to delete a row from the account table.

Stored Procedure: delete_account_confirmation

Description: This procedure is used to check whether a row in the account table can be deleted. It does so by checking if entries for bills, receipts, cheque bounce entries etc. exist for a particular account.

Stored Procedure: delete_customer _receipt

Description: This procedure is used to delete a row from the customer_receipt table.

Stored Procedure: get_blank_bill

Description: This procedure is used to get the bill numbers and their corresponding valid date ranges which are blank in the customer_bill table.

Stored Procedure: get_blank_receipt

Description: This procedure is used to get the receipt numbers and their corresponding valid date ranges which are blank in the customer_receipt table.

Stored Procedure: get_item_company

Description: This procedure is used to retrieve the values of item_company corresponding to provided values of item_type and item_name from the inventory table.

Stored Procedure: get_item_name

Description: This procedure is used to retrieve the values of item_name corresponding to provided values of item_type and item_company from the inventory table.

Stored Procedure: get_item_type

Description: This procedure is used to retrieve the values of item_type corresponding to provided values of item_name and item_company from the inventory table.

Stored Procedure: get_max_bill_no

Description: This procedure is used to return the bill number with the maximum value from the customer_bill table.

Stored Procedure: insert_account

Description: This procedure is used to insert a row into the account table

Stored Procedure: insert_customer_bill

Description: This procedure is used to insert a row into the customer_bill table.

Stored Procedure: insert_customer _bill_detail

Description: This procedure is used to insert a row into the customer_bill_detail table.

Stored Procedure: insert_customer_receipt

Description: This procedure is used to insert a row into the customer_receipt table.

Stored Procedure: is_meter_type

Description: This procedure is used to query the inventory table to identify whether a particular item is of meter type.

Stored Procedure: max_receipt_number

Description: This procedure is used to return the receipt number with the maximum value from the customer_receipt table.

Stored Procedure: transport_rate

Description: This procedure is used to return the transport rate for a particular transport from the transport_charge table.

Stored Procedure: update_account

Description: This procedure is used to update the values of a row in the account table.

Stored Procedure: update_customer_receipt

Description: This procedure is used to update the values of a row in the customer_receipt table.

Chapter 9

Code Documentation

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Namespace Documentation

Package VardhamanTextiles

Classes

- class **Account_Mantainance**
- class **bill_body**
- class **Billing**
- class **Connection**
- *This class is used for connecting frontend with MSSQL database This is used as a wrapper class all functions required for interaction with database are written here with a much easy interface database name and sql server connection name is hardcoded here itself so whenever software is deployed or run.* class **number**
- *This class is used for converting a given input figures in to word figure :an integer input word: equivalent to figures in English words.* class **Receipt**
- class **Supporter**
- class **Price_List**

  This class is used for inventory managed one can perform following actions in the inventory

Class Documentation

VardhamanTextiles.Account_Mantainance Class Reference

Member Functions

- void **Account_Mantainance_Load** (object sender, EventArgs e)

  While loading it will set position of all controls in the form.

- void **cbo_acctype_SelectedIndexChanged** (object sender, EventArgs e)

  on changing of type of account all controls will be rearragned as desired this is done by calling select function

- void **select** ()

  this procedure calls different functions based on account type as selected by customer and indicated by account type index changed

- void **customer** ()

- void **supplier** ()

  This function does the following things :-.

- void **bank** ()

  This function does the following things :- Reorders all components as per specification Rearranges tab order.

- void **employee** ()

  This function does the following things :- Reorders all components as per specification Rearranges tab order.

- void **btn_save_Click** (object sender, EventArgs e)

  whenever this button is pressed then first thing to identify is that is it called for save or update if it is called by save then appriopriote stored procedure for saving data in the back end will be done while if update button is pressed then update procedure will be called stored procedures are independent of backend no matter what backend design is stored procedure can always be made same with same number of attributes so stored procedure helps us in fixing a bond between front end and back independent of each other

- void **txt_accname_Leave** (object sender, EventArgs e)

  this will check weather their is any account matching with account name of some similar account how this works is explained in the supporter class

- void **btn_list_Click** (object sender, EventArgs e)

  on click of this button list form will be called and appripriote parameter is set in that form after that form is closed this procedure grabs the selected information and fills all the fields

- void **fill_customer** (DataTable dt)

used by above procedure for filling appropriote account type

- void **btn_delete_Click** (object sender, EventArgs e)

  used for deletion of a record from the database

- void **set_detail** (string account_type, string account_name, string account_city, string account_state)

  this is a class and is used by other forms and sets its parameter it just provides a process for adding new account and setting some fields

- void **clear** ()

  This function does the following things :- Set account type to customer clears all controls set button name and their enabled property.

- void **checkBox1_CheckedChanged** (object sender, EventArgs e)

  this decides weather to show advance entry or not

- void **txt_openbal_Leave** (object sender, EventArgs e)

  whenever open balance field is left it will be filled with appripriote precision required if combobox dosenot follows the norms then it will simply be filled with 0.00

- void **cbo_city_SelectedIndexChanged** (object sender, EventArgs e)

  it is used for filling the state field whenever user enteres the customer field

- void **Account_Mantainance_SizeChanged** (object sender, EventArgs e)
  always sets all control to the middle of screen whenever screen size is changed

---

Member Function Documentation

void VardhamanTextiles.Account_Mantainance.Account_Mantainance_Load (object **sender***, EventArgs* **e***)*

While loading it will set position of all controls in the form.

Parameters:

- *account_mantainance_form*
- *account_mantainance_form_parameter*

void VardhamanTextiles.Account_Mantainance.Account_Mantainance_SizeChanged (object **sender***, EventArgs* **e***)*

always sets all control to the middle of screen whenever screen size is changed

Parameters:

- *account_mantainance_form*
- *account_mantainance_form_parameter*

void VardhamanTextiles.Account_Mantainance.bank ()

This function does the following things :- Reorders all components as per specification Rearranges tab order.

void VardhamanTextiles.Account_Mantainance.btn_delete_Click (object **sender***, EventArgs* **e***)*

used for deletion of a record from the database

Parameters:

- *button_delete*
- *button_delete_parameter*

Account has some transaction associated with it Account have nonzero Opening balance Final Confirmation

void VardhamanTextiles.Account_Mantainance.btn_list_Click (object **sender***, EventArgs* **e***)*

on click of this button list form will be called and appripriote parameter is set in that form after that form is closed this procedure grabs the selected information and fills all the fields

Parameters:

- *button_list*
- *button_list_parameter*

void VardhamanTextiles.Account_Mantainance.btn_save_Click (object **sender***, EventArgs* **e***)*

whenever this button is pressed then first thing to identify is that is it called for save or update if it is called by save then appriopriote stored procedure for saving data in the back end will be done while if update button is pressed then update procedure will be called stored procedures are independent of backend no matter what backend design is stored procedure can always be made same with same number of attributes so stored procedure helps us in fixing a bond between front end and back independent of each other

Parameters:

- *button_save*
- *button_save_parameter*

check weather account name field is empty or not
field check based on specific error
cass backend save procedure
any other customer with same name and city dosen't exists already (for customer account) Any other account with same name dosenot exists (for bank accont) Any other account with same accountname + city + state dosenot exists (for supplier account) customer with same name but different city can exist customer with different name but same city can also exist
call backend update procedure
Whenever user want he should be able to update any field including Account Name

void VardhamanTextiles.Account_Mantainance.cbo_acctype_SelectedIndexChanged (object* sender*, EventArgs* e*)

on changing of type of account all controls will be rearragned as desired this is done by calling select function

Parameters:

- *combobox_account_type*
- *combobox_account_type_parameter*

void VardhamanTextiles.Account_Mantainance.cbo_city_SelectedIndexChanged (object* sender*, EventArgs* e*)

it is used for filling the state field whenever user enteres the customer field

Parameters:

- *combobox_city*
- *combobox_city_parameter*

void VardhamanTextiles.Account_Mantainance.checkBox1_CheckedChanged (object* sender*, EventArgs* e*)

this decides weather to show advance entry or not

Parameters:

- *checkbox_advance_entry*
- *checkbox_advance_entry_parameter*

void VardhamanTextiles.Account_Mantainance.clear ()

This function does the following things :- Set account type to customer clears all controls set button name and their enabled property.

void VardhamanTextiles.Account_Mantainance.customer ()

-This function does the following things :- -Fills city combobox -Reorders all components as per specification -Rearranges tab order

void VardhamanTextiles.Account_Mantainance.employee ()

This function does the following things :- Reorders all components as per specification Rearranges tab order.

void VardhamanTextiles.Account_Mantainance.fill_customer (DataTable* dt*)

used by above procedure for filling appropriote account type

Parameters:

- *datatabl e*

void VardhamanTextiles.Account_Mantainance.select ()

this procedure calls different functions based on account type as selected by customer and indicated by account type index changed

void VardhamanTextiles.Account_Mantainance.set_detail (string* account_type*, string* account_name*, string* account_city*, string* account_state*)

this is a class and is used by other forms and sets its parameter it just provides a process for adding new account and setting some fields

Parameters:

- *account_type*
- *account_name*
- *account_city*
- *account_state*

void VardhamanTextiles.Account_Mantainance.supplier ()

This function does the following things :-.

-Fills city combobox -Fills State Combobox -Reorders all components as per specification -Rearranges tab order

void VardhamanTextiles.Account_Mantainance.txt_accname_Leave (object* sender*, EventArgs* e*)

this will check weather their is any account matching with account name of some similar account how this works is explained in the supporter class

Parameters:

- *sender*
- *e*

customer with similar name dosen't exists already (similar can be said to be as 70% match) if exists then user should be prompted but record should be able to save

void VardhamanTextiles.Account_Mantainance.txt_openbal_Leave (object sender*, EventArgs* e*)*

whenever open balance field is left it will be filled with appripriote precision required if combobox dosenot follows the norms then it will simply be filled with 0.00

Parameters:

- *textbox_open_balance*
- *textbox_open_balance_parameter*

The documentation for this class was generated from the following file:

- File:Account Mantainance.cs

VardhamanTextiles.bill_body Class Reference

Member Functions

- void **dg_main_EditingControlShowing** (object sender, DataGridViewEditingControlShowingEventArgs e)
  This function does following things:-.
- void **dg_main_CellEndEdit** (object sender, DataGridViewCellEventArgs e)
  This function does following:-.
- void **dg_main_CellValueChanged** (object sender, DataGridViewCellEventArgs e)
  if any of the rate , quantity , meter column is item quantity , meter , rate then amount field id calculated and total , in the billing is updated using a trigger

Attributes

- TextBox **total**

Member Function Documentation

void VardhamanTextiles.bill_body.dg_main_CellEndEdit (object **sender,** *DataGridViewCellEventArgs* **e***)*

This function does following:-.
-if current edited column is any of item company, item type , item name then it fills rate corrosponding to thoes entry in the rate field -if current edited column is item quantity , meter , rate then amount field id calculated and total , in the billing is updated using a trigger

Parameters:

- *datagridview*
- *datagridview_parameter*

void VardhamanTextiles.bill_body.dg_main_CellValueChanged (object **sender,** *DataGridViewCellEventArgs* **e***)*

if any of the rate , quantity , meter column is item quantity , meter , rate then amount field id calculated and total , in the billing is updated using a trigger

Parameters:

- *datagridview*
- *datagridview_parameter*

void VardhamanTextiles.bill_body.dg_main_EditingControlShowing (object **sender,** *DataGridViewEditingControlShowingEventArgs* **e***)*

This function does following things:-.
-if current column is item company then displays a dropdown list for all company stored in the database -if current column is item type then displays a dropdown list for all type stored in the

database corrosponding to selected company -if current column is item name then it displays a drop down list for all stored item in the databasecorrosponding to selected company

Parameters:

- *datagrid*
- *datagrid_parameter*

The documentation for this class was generated from the following file:

- File:bill_body.cs

VardhamanTextiles.Billing Class Reference

Member Functions

- void **txt_billno_TextChanged** (object sender, EventArgs e)
  Whenever user changes bill number this function checks from the database weather this bill number , date combination is valid or not if this combination is valid then it does nothing but in case of invalid combination it pops out a message telling wront combination entered.
- void **FillFooter** ()
  Footer part here is refrenced to all text boxes which are involved in calculation of grand total they are.
- void **cbo_transportname_Enter** (object sender, EventArgs e)
  Whenever User enters transport name then based on transport name , customer city it checks database entry corrosponding to these information and returns sets transport charge based on above mentioned fields or else sets it to zero.
- void **recalculate_all** ()
  This function is used to recalculate following things and is used before saving Bill.
- int **check_entry** ()
  Whenever user saves bill ,before saving front end checks every entry filled is correct or not it sets appropriote flag based on these information it checks following things :-.
- int **check_bill_body** ()
  Check Bill Body function checks weather all entry in bill body are correct or not.
- int **check_bill_body_row** (int i)
  Check Bill Body Row function checks weather all entry in bill body ith row are correct or not.
- int **check_bill_rg** ()
  Check Bill RG function checks weather all entry in bill body are correct or not.
- int **check_bill_rg_row** (int i)
  Check Bill RG function checks weather all entry in bill body ith row are correct or not.
- void **btn_save_Click** (object sender, EventArgs e)
  This is used for saving whole bill in the database this checks following.

Member Function Documentation

void VardhamanTextiles.Billing.btn_save_Click (object **sender***, EventArgs* **e***)*

This is used for saving whole bill in the database this checks following.
-All critical errors are not found -in case of non critical error does user wants to still save them -inserting Bill Header and footr in the database -inserting item in the database -Inserting item

Parameters:

- *Button_refrence*
- *button_parameter*

void VardhamanTextiles.Billing.cbo_transportname_Enter (object **sender***, EventArgs* **e***)*

Whenever User enters transport name then based on transport name , customer city it checks database entry corrosponding to these information and returns sets transport charge based on above mentioned fields or else sets it to zero.

Parameters:

- *combobox_transport_name*
- *combobox_transport_name_parameter*

int VardhamanTextiles.Billing.check_bill_body ()

Check Bill Body function checks weather all entry in bill body are correct or not.

int VardhamanTextiles.Billing.check_bill_body_row (int **i***)*

Check Bill Body Row function checks weather all entry in bill body ith row are correct or not.

Parameters:

- *row_index* Ith index of the bill body table

int VardhamanTextiles.Billing.check_bill_rg ()

Check Bill RG function checks weather all entry in bill body are correct or not

int VardhamanTextiles.Billing.check_bill_rg_row (int **i***)*

Check Bill RG function checks weather all entry in bill body ith row are correct or not.

int VardhamanTextiles.Billing.check_entry ()

Whenever user saves bill ,before saving front end checks every entry filled is correct or not it sets appropriote flag based on these information it checks following things :-.
-Validity of customer name -Validity of date , bill number combination -validity of all items entered (its over all item name , item quantity , item price are properly filled or not and weather amount is properly calculated or not -changing row color on finding of error -recalculation of amount -appending appropriote zeros and rounding off of rate and amount -checking all these entries for rg also -recalculating footer information -if every things mentioned abobe are done correctly then it returns true -if critical things are not proper like customer name then it returns 0 -if non critical error are identified then it returns -1
customer name check condition

void VardhamanTextiles.Billing.FillFooter ()

Footer part here is refrenced to all text boxes which are involved in calculation of grand total they are.
-rg total -total -cash payment -CD/EXP -Expper -Transport Total it calculates grand total based on all these fields

void VardhamanTextiles.Billing.recalculate_all ()

This function is used to recalculate following things and is used before saving Bill.
-Amount -Footer Information --Expenses --Grandtotal

void VardhamanTextiles.Billing.txt_billno_TextChanged (object **sender***, EventArgs* **e***)*

Whenever user changes bill number this function checks from the database weather this bill number , date combination is valid or not if this combination is valid then it does nothing but in case of invalid combination it pops out a message telling wront combination entered.

Parameters:

- *Textbox refrence*
- *Textbox_reference_property*

The documentation for this class was generated from the following file:

- File:Billing.cs

VardhamanTextiles.Connection Class Reference

This class is used for connecting frontend with MSSQL database This is used as a wrapper class all functions required for intraction with database are written here with a much easy interface database name and sql server connection name is hardcoded here itself so whenever software is deployed or run.

Member Functions

- **Connection** ()
  this is a constructor and is used to initialize connection string it sets all required datamember of connection string if their is any error in the connection string then exception occurs which pops some message about failure
- SqlConnectionStringBuilder **connectionstring** ()
  this is used to return the connection string this is usefull if someone wants to use some functionality which this class donot does
- bool **connent** ()
  This function is used for creation of a connection pool in the sql server database a request for connection is made and if success then returns true and connects with the database but if its not a sunnccess then it returns false and pops error message about the failure.
- bool **disconnect** ()
  This function is used to disconnect the already created connection but returns an error when connection cannot be disconnected.
- DataTable **getTable** (String qurey)
  This class takes a query to be executed in the database and terurns outpot of that query in the form of datatable.
- int **exeNonQurey** (String qurey)
  this function takes a query , executes it on the existing connection and returns weather query was a success or not if it was a success then returns integer 1 otherwise returns integer 0 this returns error when qither query is incorrect or connection is in closed state
- DataSet **dsentry** (string query, string table)
  This class takes a query to be executed in the database and terurns outpot of that query in the form of dataset.
- SqlDataReader **exereader** (String qurey)
  This procedure execured a query query execution results in a table generation at the end datareader reads this datatable one by one by each time querying the database about data of a perticular column this returns error when qither query is incorrect or connection is in closed state.
- void **closereader** ()
  as datareader reads data one by one each time quering from the database so it is left opened until required data is fetched this function gives us ability to close thoes reader when its work is over this functions returns error when datareader is already closed
- string **exesclr** (String qurey)
  when query output contains only a single data then this procedure is used instead of getting the datatable or dataset this is much better approach when query output contains only single output this returns error when qither query is incorrect or connection is in closed state

Protected Attributes

- SqlConnection **conn**

Detailed Description

This class is used for connecting frontend with MSSQL database This is used as a wrapper class all functions required for intraction with database are written here with a much easy interface database name and sql server connection name is hardcoded here itself so whenever software is deployed or run.

Constructor & Destructor Documentation

VardhamanTextiles.Connection.Connection ()

this is a constructor and is used to initialize connection string it sets all required datamember of connection string if their is any error in the connection string then exception occurs which pops some message about failure

---

Member Function Documentation

void VardhamanTextiles.Connection.closereader ()

as datareader reads data one by one each time quering from the database so it is left opened until required data is fetched this function gives us ability to close thoes reader when its work is over this functions returns error when datareader is already closed

SqlConnectionStringBuilder VardhamanTextiles.Connection.connectionstring ()

this is used to return the connection string this is usefull if someone wants to use some functionality which this class donot does

Returns:

Sql **Connection** String

bool VardhamanTextiles.Connection.connent ()

This function is used for creation of a connection pool in the sql server database a request for connection is made and if success then returns true and connects with the database but if its not a sunnccess then it returns false and pops error message about the failure.

Returns:

`True` if connected<c>falseif connection failed

bool VardhamanTextiles.Connection.disconnect ()

This function is used to disconnect the already created connection but returns an error when connection cannot be disconnected.

-this perticularly happens when some data transfer is already in progress -when connection is already been disconnected

Returns:

`True` If disconnection is successfull `False` If it operation failed

DataSet VardhamanTextiles.Connection.dsentry (string query*, string* table*)*

This class takes a query to be executed in the database and terurns outpot of that query in the form of dataset.
-it returns error when bad query is inputted -it retuerns error when connection is in closed state this returns error when qither query is incorrect or connection is in closed state

Parameters:

- *query*
- *table*

int VardhamanTextiles.Connection.exeNonQurey (String qurey*)*

this function takes a query , executes it on the existing connection and returns weather query was a success or not if it was a success then returns integer 1 otherwise returns integer 0 this returns error when qither query is incorrect or connection is in closed state

Parameters:

- *qurey*

SqlDataReader VardhamanTextiles.Connection.exereader (String qurey*)*

This procedure execured a query query execution results in a table generation at the end datareader reads this datatable one by one by each time querying the database about data of a perticular column this returns error when qither query is incorrect or connection is in closed state.

Parameters:

- *qurey*

string VardhamanTextiles.Connection.exesclr (String qurey*)*

when query output contains only a single data then this procedure is used instead of getting the datatable or dataset this is much better approach when query output contains only single output this returns error when qither query is incorrect or connection is in closed state

Parameters:

- *qurey*

DataTable VardhamanTextiles.Connection.getTable (String qurey*)*

This class takes a query to be executed in the database and terurns outpot of that query in the form of datatable.

-it returns error when bad query is inputted -it retuerns error when connection is in closed state

Parameters:

- *qurey*

The documentation for this class was generated from the following file:

- File:Connection.cs

VardhamanTextiles.number Class Reference

This class is used for converting a given input figures in to word figure:an integer input word:equivalent to figures in english words.

Static Member Functions

- static string **num2text** (Int32 value)
  given an input word in value it outputs an english sentence on a way it is pronounced
- static string **text** (Int32 x)
  all input combinations using whome we can create any number between One and Hundred

---

Detailed Description

This class is used for converting a given input figures in to word figure:an integer input word:equivalent to figures in english words.

Member Function Documentation

static string VardhamanTextiles.number.num2text (Int32 **value***) [inline, static]*

given an input word in value it outputs an english sentence on a way it is pronounced

Parameters:

- *value*

static string VardhamanTextiles.number.text (Int32 **x***) [inline, static]*

all input combinations using whome we can create any number between One and Hundred

Parameters:

- *value*

The documentation for this class was generated from the following file:

- File:number.cs

VardhamanTextiles.Price_List Class Reference

This class is used for inventory managed one can perform following actions in the inventory:-.

Member Functions

- void **Price_List_Load** (object sender, EventArgs e)
  when this form is opened then List of all inventory are captured updated in the list view
- void **cbo_rate_Enter** (object sender, EventArgs e)
  based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value
- void **cbo_company_Enter** (object sender, EventArgs e)
  based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value
- void **cbo_item_type_Enter** (object sender, EventArgs e)
  based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value
- void **cbo_item_name_Enter** (object sender, EventArgs e)
  based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value
- void **dataGridView1_DoubleClick** (object sender, EventArgs e)
  when user doubleclicks in any entry in the datagridview then all its firlds are filled in the above form fields and user is now also allowed to update
- void **btn_save_Click** (object sender, EventArgs e)
  This checks following :-.

Detailed Description

This class is used for inventory managed one can perform following actions in the inventory:-.
-Add -Update -Delete -Quantity Management

Member Function Documentation

void VardhamanTextiles.Price_List.btn_save_Click (object sender*, EventArgs* e*)*

This checks following :-.
-All entries are filled correctly or not

Parameters:

- *save_button*
- *save_button_parameter*

void VardhamanTextiles.Price_List.cbo_company_Enter (object sender*, EventArgs* e*)*

based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value

Parameters:

- *combobox_company*
- *combobox_company_parameter*

void VardhamanTextiles.Price_List.cbo_item_name_Enter (object **sender***, EventArgs* **e***)*

based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value

Parameters:

- *combobox_item_name*
- *combobox_item_name_parameter*

void VardhamanTextiles.Price_List.cbo_item_type_Enter (object **sender***, EventArgs* **e***)*

based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value

Parameters:

- *textbox_item_type*
- *textbox_item_type_parameter*

void VardhamanTextiles.Price_List.cbo_rate_Enter (object **sender***, EventArgs* **e***)*

based on all entries which user has entered this function will update list based on item company , item type this also updates all combobox allowable value

Parameters:

- *combobox_rate*
- *combobox_rate_parameter*

void VardhamanTextiles.Price_List.dataGridView1_DoubleClick (object **sender***, EventArgs* **e***)*

when user doubleclicks in any entry in the datagridview then all its firlds are filled in the above form fields and user is now also allowed to update

Parameters:

- *datagridview*

- *datagridview_parameter*

void VardhamanTextiles.Price_List.Price_List_Load (object **sender***, EventArgs* **e***)*

when this form is opened then List of all inventory are captured updated in the list view

Parameters:

- *Form*
- *Form_parameter*

The documentation for this class was generated from the following file:

- File:Price List.cs

VardhamanTextiles.Receipt Class Reference

Member Functions

- void **Receipt_Load** (object sender, EventArgs e)
  at form load time allowed receipt number is automatically filled in the receipt area , date time is set of todays date receipt number is setted as max receipt number + 1 where max is zero if no receipt is made tll now
- void **chk_bank_CheckedChanged** (object sender, EventArgs e)
  Whenever user wants to enter cheque details then is bank entry check box has to be ticked when this is ticked then this function creates an interface where cheque entry can be done.
- void **btn_save_Click** (object sender, EventArgs e)
  *Whenever user clicks save button followni*g *things are being done :-.*
- void **txt_amount_Leave** (object sender, EventArgs e)
  when user finishes writing amount foeld then this function is invoked and sets appropriote zeros and rounding off to amount field this functions also does following :-
- void **txt_cd_Leave** (object sender, EventArgs e)
  when user finishes writing cash discount foeld then this function is invoked and sets appropriote zeros and rounding off to amount field this functions also does following :-
- void **txt_receiptno_Leave** (object sender, EventArgs e)
  Whenever user leaves this box our database checks weather this receipt number , date combination is correct or not if they are not appropriote then user is shown a popup box showing valid receipt number , date combination.
- void **btn_update_Click** (object sender, EventArgs e)
  Whenever user selects something from list then update and delete buttons gets enabled when user presses this button then it checks.
- void **btn_delete_Click** (object sender, EventArgs e)
  Whenever user selects some item from the list then this button gets enabled and user is allowed to delete this entry on deletion database dletes this receipt i.e. it simply deletes all entry pertaning to this receipt number.
- void **btn_list_Click** (object sender, EventArgs e)
  When user clicks this button then a popup is shown to the user where user can select any one receipt in this list user can also search once user double ck=loicks on any one of the entry then all fields in the form are automatically filled based on the user selected receipt number.
- void **set_field_from_list** (string receipt_no, string date, string customer_name, string customer_city, string amount, string cash_discount, string total, string bill_no, string note, string bank_name, string ch_dd_no, string ch_date, string ch_due_date)
  This function is used to fill receipt details on doubleclicking of the user for any entry in the list.

Member Function Documentation

void VardhamanTextiles.Receipt.btn_delete_Click (object sender*, EventArgs* e*)*

Whenever user selects some item from the list then this button gets enabled and user is allowed to delete this entry on deletion database dletes this receipt i.e. it simply deletes all entry pertaning to this receipt number.

Parameters:

- *button_delete*
- *button_delete_parameter*

void VardhamanTextiles.Receipt.btn_list_Click (object **sender***, EventArgs* **e***)*

When user clicks this button then a popup is shown to the user where user can select any one receipt in this list user can also search once user double ck=loicks on any one of the entry then all fields in the form are automatically filled based on the user selected receipt number.

Parameters:

- *button_list*
- *button_list_parameter*

void VardhamanTextiles.Receipt.btn_save_Click (object **sender***, EventArgs* **e***)*

Whenever user clicks save button followni things are being done :-.
-Checks weather or not all entries are correctly filled or not -Checking receipt number , date combination is correct or not After this a query is send to the database and based on the output of database query this function pops out an error message box

Parameters:

- *savebutton*
- *save_button_parameter*

void VardhamanTextiles.Receipt.btn_update_Click (object **sender***, EventArgs* **e***)*

Whenever user selects something from list then update and delete buttons gets enabled when user presses this button then it checks.
-all entries are valid or not -receipt number , date combination are correct or not once validation are done then these written entries are updated in the database based on the database returned values popup message box are shown to the user

Parameters:

- *button_update*
- *button_update_parameter*

void VardhamanTextiles.Receipt.chk_bank_CheckedChanged (object **sender***, EventArgs* **e***)*

Whenever user wants to enter cheque details then is bank entry check box has to be ticked when this is ticked then this function creates an interface where cheque entry can be done.

Parameters:

- *checkbox*

- *checkbox_parameter*

void VardhamanTextiles.Receipt.Receipt_Load (object **sender**, *EventArgs* **e***)*

at form load time allowed receipt number is automatically filled in the receipt area , date time is set of todays date receipt number is setted as max receipt number + 1 where max is zero if no receipt is made tll now

Parameters:

- *Receipt_Form*
- *Receipt_Form_Parameter*

void VardhamanTextiles.Receipt.set_field_from_list (string **receipt_no**, *string* **date**, *string* **customer_name**, *string* **customer_city**, *string* **amount**, *string* **cash_discount**, *string* **total**, *string* **bill_no**, *string* **note**, *string* **bank_name**, *string* **ch_dd_no**, *string* **ch_date**, *string* **ch_due_date***)*

This function is used to fill receipt details on doubleclicking of the user for any entry in the list.

Parameters:

- *receipt_no*
- *date*
- *customer_name*
- *customer_city*
- *amount*
- *cash_discount*
- *total*
- *bill_no*
- *note*
- *bank_name*
- *ch_dd_no*
- *ch_date*
- *ch_due_date*

void VardhamanTextiles.Receipt.txt_amount_Leave (object **sender**, *EventArgs* **e***)*

when user finishes writing amount foeld then this function is invoked and sets appropriote zeros and rounding off to amount field this functions also does following :-
-calculating total based on amount and cash discount -filling words to figure

Parameters:

- *amount_textbox*
- *amount_textbox_parameter*

void VardhamanTextiles.Receipt.txt_cd_Leave (object **sender***, EventArgs* **e***)*

when user finishes writing cash discount foeld then this function is invoked and sets appropriote zeros and rounding off to amount field this functions also does following :-
-calculating total based on amount and cash discount -filling words to figure

Parameters:

- *cash_discount_textbox*
- *cash_discount_textbox_parameter*

void VardhamanTextiles.Receipt.txt_receiptno_Leave (object **sender***, EventArgs* **e***)*

Whenever user leaves this box our database checks weather this receipt number , date combination is correct or not if they are not appropriote then user is shown a popup box showing valid receipt number , date combination.

Parameters:

- *receipt_number_textbox*
- *receipt_number_textbox_parameter*

The documentation for this class was generated from the following file:

- File:Receipt.cs

Chapter 10

Test Case Summary

Test Case

Table Summary

ID	Name	Description	State
31	Validate summary report for a valid user	Validate summary report functionality for a valid user	Draft
36	Billing Header Validation for existing customer	This test case will verify the "Customer Name" and "City" fields entry and functionality of "Add/Update" button in billing form.	Draft
37	Billing Header Validation for New Customer	This test case will verify the functionality of "Add/Update" button when a new Customer Name is entered in "Customer Name" field.	Draft
38	Billing Header Validation for "Through", "Date", "Bill No" Field	This test case will verify the functionality of "Through", "Date", & "Bill No" button in billing form.	Draft
39	Test case_in Validation of Receipt form.	This test case will verify the "Customer Name" and "City" fields entry and functionality of "Add/Update" button in receipt form.	Draft
40	Billing Footer Validation with customer account existing in database.	This test case will verify the functionality of "save" "Clear" "Print" and "Print all" buttons in the billing footer.	Draft
41	Billing Footer Validation "save" button with new customer	This test case will verify the functionality of save button in case of saving a bill for a new cutomer	Draft
42	Billing Footer Validation "Clear" Button	This test case will verify the functionality of "Clear" button in the filling footer.	Draft
43	Billing Footer validation "Save Print " button for existing customer.	This test case will verify functionality of"Save Print" button for an existing customer.	Draft
44	Billing footer validation- "Print All" button	This test case will verify functionality of" Print All" button for an existing customer.	Draft
45	Billing Footer validation: "Save Print" button with new customer	This test case will verify functionality of"Save Print" button for a new customer.	Draft
46	Billing Footer validation: "Print All" button for new customer	This test case will verify functionality of"Print All" button for a new customer.	Draft
47	Billing Footer Validation- "Check Item" button	This test case will verify the functionality of check item button before the bill is finally saved or printed.	Draft
61	Test Case for List form	Test Case for List form	Draft
62	Test case for customer Account Maintenance form "Save Print" Button	Test case for customer Account Maintenance form "Save Print" Button	Draft
63	Test case for customer Account Maintenance form "Save " Button	Test case for customer Account Maintenance form "Save " Button	Draft
64	Test case for supplier Account Maintenance form "Save " Button	Test case for supplier Account Maintenance form "Save " Button	Draft
65	Test case for supplier Account Maintenance form "Save Print " Button	Test case for supplier Account Maintenance form "Save Print " Button	Draft
66	Test case for bank Account Maintenance form "Save Print " Button	Test case for bank Account Maintenance form "Save Print " Button	Draft
67	Test case for bank Account Maintenance form "Save" Button	Test case for bank Account Maintenance form "Save" Button	Draft
68	Test case for employee Account Maintenance form "Save" Button	Test case for employee Account Maintenance form "Save" Button	Draft
69	Test case for employee Account Maintenance form "Save Print" Button	Test case for employee Account Maintenance form "Save Print" Button	Draft
70	validation of "Clear" Button of Account Maintenance form	validation of "Clear" Button of Account Maintenance form	Draft
71	Test case to validate Deafult functioanlity and GUI of for Account Maintenance form	Test case to validate Deafult functioanlity and GUI of for Account Maintenance form	Draft
73	For checking the link of Receipt Form	For checking the link of Receipt Form	Draft
74	For validating the format of Receipt form	For validating the format of Receipt form	Draft
75	For validating the save button on receipt form	For validate the save button on receipt form	Draft
76	For validating the Add button on receipt form	For validating the Add button on receipt form	Draft
77	For validating the Save and Print button on receipt form	For validating the Save and Print button on receipt form	Draft
78	For validating the delete button on receipt form	For validating the delete button on receipt form	Draft

Chapter 11

Cost Estimation

Man Hour Rate	100/- per hour
Number of members in the project	08
Number of working hours per day	03
Number of working days per week	04
Total Number of weeks required for completion	07
Total Cost of Development of Project	RS. 67200/-

Chapter 12

Installation Manual

11.1 Working Environment

- Windows XP
- 80 GB Hard Disk
- 256 MB Ram
- Printer
- 1024x786 Maximum screen resolution

11.2 Installation Manual

- Copy software anywhere in the hard disk
- Install visual c# 2008 trial version
- Install Sql Server 2005 express version
- Install Sql Server management studio express
- Install crystal reports
- Open Sql server management studio express
- Connect->Database->Rightclick->Attach
- Attach db_vatdhamantextile , db_address , db_common these three files
- Press OK
- If Sql server connection name is not sqlserver then open visual c#.Net , in solution explorer go to connection , change sql server name , from ./sqlexpress to ./server name<> , build solution
- Go to copied folder and browse bin folder

- Exe file is the overall file , user can now run software without any other setting

Chapter 13

Quick Start Guide

1. Go to start Menu and type Vardhaman.exe
2. Click on Vardhaman.exe
3. Main Form will get displayed
4. Click on ‘Customer’ in Menu Bar
    I. Select ‘Account Maintenance’ to create or update a customer account.
    II. Select ‘Billing’ to create or edit a customer bill.
    III. Select ‘Receipt’ to create or edit a customer receipt.
    IV. Select ‘Ledger’ to create, edits, or view a customer account detail.
    V. Select ‘Cheque Bounce Entry’ to enter any cheque bounce incidence against some customers.
5. Click on ‘Supplier’ in Menu Bar
    I. Select ‘Account Maintenance’ to create or update a supplier account.
    II. Select ‘Bill’ to create or edit a supplier bill.
    III. Select ‘Payment’ to create or view the detail of any payment made against any supplier.
    IV. Select ‘Receipt’ to create or edit a supplier receipt.
6. Click on ‘Bank’ in Menu Bar
    I. Select ‘Account Maintenance’ to create or edit any record of bank transaction made against any customer, Supplier, or Employee
7. Click on ‘Employee’ in Menu Bar

I. Select ‘Account Maintenance’ to create or edit an employee account.

II. Select ‘Attendance’ to mark Absent/Present against any employee.

8. Click on ‘Price List’ in Menu Bar to view or enter the price detail of any item.
9. Click on ‘Backup’ in Menu Bar to create Backup of Database instance with unique identification.
10. Click on ‘Restore’ in Menu Bar to restore the database on some previous restore point.
11. Click on ‘Phone Directory’ in Menu Bar to enter, edit, or view any contact information.

Chapter 14

Glossary

Bales	A large package of raw or finished material tightly bound with twine or wire and often wrapped
Broker	An individual or firm employed by others to plan and organize sales or negotiate contracts for a commission
Commission	A fee charged by a broker or agent for his/her service in facilitating a transaction
Customer	One who regularly or repeatedly makes purchases of a trader
Discount	The sale of goods in large quantities, as for resale by a retailer
Ledger	An accounting book of final entry where transactions are listed in separate accounts
Liabilities	An obligation that legally binds an individual or company to settle a debt
Negotiate	To confer with another or others in order to come to terms or reach an agreement
Retailer	A business or person that sells goods to the consumer, as opposed to a wholesaler or supplier
Supplier	A company which supplies parts or services to another company
Wholesale	The sale of goods in large quantities, as for resale by a retailer