Backend Development

# Backend Tables/Objects

## Customer

Customer ID – Auto-Generated By backend

Customer Name - Passed to your method from front end

Customer City: Passed to your method from front end (This can also be used as an area within the same city)

Org Id: Passed to your method from front end

CreatedByID: ( This is to track which employee id is used as supervisor id)

GSTIN (String: 20): GST Number of the customer

Transporter (String:256): name and address of the transport company for the customer.

Mobile Number1 (15 char):

Mobile Number2 (15 char):

Firm Name (String:256): Firm/shop/Business Name

Billing Adress (String: 256): Full address on which to be billed.

**Operation Required:**

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Input (mandatory)** | **Description** |
| Create  (with transaction enabled with “Account”) | Customer Name,  Customer City,  Mobile Number,  Org ID | 1. Create a customer in the database.  If Firm Name is null then copy Customer Name here.  2. Create Account  **Account NickName Generation Logic:**  Append “Customer Name” and “Customer Firm Name”  If Customer Name == Firm Name then  Account NickName = “[City Name] Firm Name”  ELSE  Account NickName = “[<Customer City] <Firm Name> - <Customer Name>”  **Owner ID**  Customer ID  **Type**  “Customer” |

Create, Update, Search (Based on Name and City)

ToDO: Bring dues with the customer list

Customer Creation: Whenever a new customer is created, an account will be created in an account table. Customer Name and Customer Firm Name would be appended to create an account name.

## Delivery

Delivery Date: Passed to your method from front end

Customer ID: Passed to your method from front end

Description: Passed to your method from front end

Org Id: Passed to your method from front end

CreatedByID: ( This is to track which employee id is used as supervisor id)

**Operation Required:**

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create | Delivery Date  Customer ID  Org Id  CreatedByID | 1. Create an Account in the database. |
| Search | Org ID |  |

## CustomerInvoice

Invoice means when we bill to our customer. Meaning when we are selling something to other party.

ToCustomer ID: Automatically calculated from the given “ToAccountID”. The ‘ToCustomerID” will be get from “account:owner” from given “ToAccountID”.

Date: Passed to your method from front end

Order Amount (float) : Passed to your method from front end

CGST (float):

SGST(float):

IGST(float):

EXTRA(float):

IncludeInCalc: True/False

Total (float): Passed to your method from front end

Org Id: Passed to your method from front end

CreatedByID: ( This is to track which employee id is used as supervisor id)

Notes (String:256): General Notes

IncludeInReport: True / false (flag indicating whether to include in GST report)

BillNO: (Text) :

ToAccountID: From UI.

**Operation Required:**

Create, Search

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create  (In Transaction with UpdateAccount) | Org Id,  Customer ID,  Date,  Total Amount,  CreatedByID,  Account ID | 1.Create a bill  2. Whenever a bill is created, the account of the other-party (customer/vendor) is updated and it’s current balance is subtracted with Total Amount.  (e.g Current Balance = -20,000 (i.e debt of 20,000 on customer/vendor)  Now, A bill is created with amount=5000  So new updated balance for that party should be: -20,000 – 5000 = -25000)    Update Account (call Account:UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:Account ID”  CurrentBalance : pass ”parameter:Total Amount”  Operation=SUBTRACT  3. Update “CalculationTracker” |
| Search | Org ID |  |
|  |  |  |

## General Expense

This table primarily tracks the expenses where we do not

Expense ID: Auto Generated by backend

Expense Date: Passed to your method from front end

Expense Category: Free Text ! Passed to your method from front end

Expense Description: Free Text ! Passed to your method from front end

Expense Amount (float) : Passed to your method from front end

To Party Name: Free Text ! Passed to your method from front end

Org Id: Passed to your method from front end.

CreatedByID: ( This is to track which employee id is used as supervisor id)

FromAccountID (String:30) : This will be passed.

ConsiderInCalc: True/False

**FromEmployeeID**: Automatically calculated from “FromAccountID”. Query account table for “owner”

**Operation Required:** Create

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create  (In Transaction with Account:UpdateAccountBalance  CalculationTracker:UpdatePaidExpense or UpdateUnpaidExpense) | Org Id,  Date,  Expense Amount,  CreatedByID,  Account ID | 1.Create an expense in the database table.  2. Whenever a “paid” expense is created, the account of the paying party (our employee) is updated and it’s current balance is subtracted with Expense Amount.  (e.g Current Employee Balance = 10,000  Now, an expense is created with amount= 2,000  So new updated balance for that employee should be: 10,000 - 2000 = 8000)    A positive balance represent that we have to give (orgnization id) that balance to other party.  Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:Account ID”  CurrentBalance : pass ”parameter:Expense Amount”  Operation=SUBTRACT  3.  If “paid=false” then:  Call updateUnpaidExpenseBalance(Operation) method  With “Operations=ADD”  If “paid=true” then:  Call updatePaidExpenseBalance(Operation) method  “Operation=ADD” |
| updateUnpaidToPaidExpense | ExpenseID  Amount | 2. Whenever an expense becomes “paid” from “Unpaid”, the account of the paying party (our employee) is updated and it’s current balance is subtracted with Expense Amount. We need to subtract from unpaidExpense and add into the PaidExpense in **CalculationTracker** table.  Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:Account ID”  CurrentBalance : pass ”parameter:Expense Amount”  Operation=SUBTRACT  3.  Call updateUnpaidExpenseBalance(Operation) method  With “Operations=SUBTRACT”  Call updatePaidExpenseBalance(Operation) method  With “Operation=ADD” |
| Search | Org ID |  |

## PaymentReceived

Payment will be received from a customer to a employee account.

PaymentRcvdID: Auto Generated by Backend

FromAccountID (String:20): (Referece to Account ID )

RcvdDate: Received Date

CreatedByID: ( This is to track which employee id is used as supervisor id)

ToAccountID(String:20): Reference to the Account Table.

Amount (float):

**FromCustomerID**: This will be auto caluclated from the “FromAccountID” field. You need to query account table for the owner.

**ToEmployeeID**: This will be auto caluclated from the “ToAccountID” field. You need to query account table for the owner.

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create  (In Transaction with UpdateAccount) | Org Id,  Customer ID,  Date,  Amount,  CreatedByID,  From Account ID  To Account ID | 1.Create a payment Received in the database  2. Whenever a payment is created, the account of the paying-party (customer/vendor: from Account) is updated and it’s current balance is subtracted with “Amount” Paid.  (e.g Current Balance = -20,000 (i.e debt of 20,000 on customer/vendor)  Now, the customer has made a payment of: 15000  So new updated balance for that customer should be:  -20,000 + 15000 = -5000)    Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:From Account ID”  CurrentBalance : pass ”parameter:Amount”  Operation=ADD  3. Whenever a payment is created, the account of the receiving-party (Our Employee: To Account) is updated and it’s current balance is added with “Amount” received.  (e.g Current Employee Balance = 20,000  Now, the customer has made a payment of: 15000  So new updated balance for that customer should be:  20,000 + 15000 = 35000)    Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:To Account ID”  CurrentBalance : pass ”parameter:Amount”  Operation=ADD |
| Search | Org ID |  |

## Account

An employee can have multiple accounts, a vendor can have multiple accounts so we have introduced a concept of accounts. Please note, this will work as “toAccount” or it will be an internal account. This can be a real bank account e.g company current Account. Whenever an employee is created, an associated named account will be created. In the first release, Account will not get created from UI, it will be created as part of the “create-customer” business logic.

**AccountID**: Auto-Generated by the backend

**OwnerID**\*: Owner of the account (CustomerId or EmployeeID)

**AccountNickName**\*: (Text: 300) A simple identifiable description like AviICICI, AviSBI, Nilesh PNB, Company Current Account (PNB), Company Current Account (ICICI)

Currently, this would be created using Customer’s Name appended with firm Name.

**Details** (String:100): (here we can select what is the details of the accounts like Account Number, UPI etc)

**Type**\*: **Customer** or **Employee**: (This will be used when doing a “join” operation with either with employee or customer table)

**CurrentBalance**: (Number): This field will be used for maintaining a current balance of an account at any given time. This will be used for entering the initial balance during customer/vendor/employee creation.

**LastBalance**: (Number): This field will maintain the history of the “CurrentBalance” field. Before updating the CurrentBalance field, the value of “CurrentBalance” will be copied here.

Org Id: Organization ID

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create | AccountNickName,  Owner ID,  Type | 1. Create an Account in the database. |
| UpdateAccountBalance | Connection,  AccountID,  Balance  Operation(ADD,SUBSTRACT) | This will run in the transaction with calling method. For this given AccountID:  Copy value of table:“CurrentBalance” to table: “LastBalance”    NewBalance = Operation (“Parameter: Balance” , “table:CurrentBalance” )  Update “Variable:New Balance” to “table:CurrentBalance” |
| Search | Org ID |  |

## PurchasefromVendor

This is to track when a material is purchased from a vendor.

Purcahse\_id: Auto genearted ID

FromCustomerId: This will be auto-caculated from the “FromAccountID”. For this, get “Owner” from “Account” Table with given “FromAccountID”.

Bill\_ID: (free text) : Actual Bill ID if available.

Net Amount (float):

CGST ((float)) :

SGST (float) :

Extra (float) :

Total Amount (float):

Material Details: (TextField)

ConsiderInCalc: True/False

IncludeInReport:True/False

EXTRA(float): Number field

FromAccountID: This will be passed from frontend.

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create  (In Transaction with Account:UpdateAccountBalance ) | Org Id,  Customer ID,  Date,  Total Amount,  CreatedByID,  Account ID | 1.Create a Invoice  2. Whenever a bill is created, the account of the other-party (customer/vendor) is updated and it’s current balance is subtracted with Total Amount.  (e.g Current Balance = 10,000 (i.e debt of 20,000 on customer/vendor)  Now, A invoice is created with amount=5000  So new updated balance for that party should be: -10,000 + 5000 = 15000)    A positive balance represent that we have to give (orgnization id) that balance to other party.  Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:Account ID”  CurrentBalance : pass ”parameter:Total Amount”  Operation=SUBTRACT |
| Search | Org ID |  |

## PayToVendor

PayToVendorID: Auto Generated ID

Date:

ToCustomerId: Auto-calculated from the given “ToAccountID”. Get “Owner” from the “account” table with given “ToAccountID”

Amount: (float)

FromAccountID: From Account ID linked to Account Table (Employee).

ToAccountID:To Account ID linked to Account Table (Customer/Vendor). Passed from UI.

EmployeeID: Link to Employee Table.

IncludeInCalc: True/False

Details:

CreatedByID: ( This is to track which employee id is used as supervisor id)

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create  (In Transaction with UpdateAccount) | Org Id,  Customer ID,  Date,  Amount,  CreatedByID,  From Account ID  To Account ID | 1.Create a “payment made” in the database  2. Whenever a payment is made by our employee, the account of the other party (vendor: To Account) is updated and it’s current balance is subtracted with “Amount” Paid.  (e.g Current Balance Of Vendor = 20,000 (i.e debt of 20,000 on us)  Now, the we have made a payment of: 15000  So new updated balance for that vendor should be:  20,000 - 15000 = 5000)  Here postive balance represent that we have to give that money to vendor/customer.  Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:To Account ID”  CurrentBalance : pass ”parameter:Amount”  Operation=SUBTRACT  3. Whenever a payment is made, the account of the paying-party (Our Employee: From Account) is updated and it’s current balance is subtracted with “Amount” paid.  (e.g Current Employee Balance = 20,000  Now, the employee has made a payment of: 15000  So new updated balance for that customer should be:  20,000 - 15000 = 5000)    Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:From Account ID”  CurrentBalance : pass ”parameter:Amount”  Operation=SUBTRACT |
| Search | Org ID |  |

## InternalTransfer

InternalTransferID: Auto Generated ID

TransferDate:

Amount: (float):

FromAccountID: Account ID link to Account Table.

FromEmployeeID: Employee Id linked to Employee Table.

ToAccountId: Account ID link to Account Table.

ToEmployeeID: Employee Id linked to Employee Table.

IncludeInCalc: True/False

Details:

CreatedByID: ( This is to track which employee id is used as supervisor id)

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create  (In Transaction with UpdateAccount) | Org Id,  Customer ID,  Date,  Amount,  CreatedByID,  From Account ID  To Account ID | 1.Create a “Internal Transfer” in the database  2. Whenever a payment is made by our employee to other employee, the account of the other/receiving party (ToAccountID) is updated and it’s current balance is added with “Amount” Paid.  (e.g Current Balance Of Receiving Employee = 20,000  Now, a payment is made of : 15000  So new updated balance for that receiving employee should be:  20,000 + 15000 = 35000)  Here postive balance represent that he has that much of money with him.  Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:To Account ID”  CurrentBalance : pass ”parameter:Amount”  Operation=ADD  3. Whenever a payment is made, the account of the paying-party/employee(FromAccountID) is updated and it’s current balance is subtracted with “Amount” paid.  (e.g Current Employee Balance = 20,000  Now, the employee has made a payment of: 15000  So new updated balance for that customer should be:  20,000 - 15000 = 5000)    Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:From Account ID”  CurrentBalance : pass ”parameter:Amount”  Operation=SUBTRACT |
| Search | Org ID |  |

## Employee

Employee tracks the people working for an organization. These are internal personnel working in a business/organization/firm.

* Employee ID (Auto Generated by backend)
* First Name
* Last Name
* Mobile Number
* loginRequired (boolean)
* loginUserName:
* loginPassword
* Org Id: Passed to your method from front end
* Type: (Text)
* CurrentSalaryBalance
* PreviousSalaryBalance (maintain history for CurrentSalaryBalance field)

If a loginRequired is true then user will be created with a 8 digit password and sent on the Mobile Number Provided.

Currently we are not modelling roles, we are going with the approach of user login in or not login in into the app. If a user require login then he/she will be able to see the whole view.

**Operation Required:** Create

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create | First Name,  Last Name,  Mobile Number  Org Id | 1. Create an Employee in the database. |
| UpdateEmpSalaryBalance | Connection,  EmployeeID,  Balance  Operation(ADD,SUBSTRACT) | This will run in the transaction with calling method. For this given EmployeeID:  Copy value of table:“CurrentSalaryBalance” to table: “LastBalance”    NewBalance = Operation (“Parameter: Balance” , “table:CurrentBalance” )  Update “Variable:New Balance” to “table:CurrentBalance” |
|  |  |  |

## EmployeeSalary

This tables tracks the Salary dues. The Salary due is to track how much salary, perks or benefit to be given to the employee. For Example – a person gets 5000/- salary for the month of March 2020 but due to lockdown you have not paid it. So we need to maintain the balance amount.

SalaryDueID: Auto Generated ID

Amount: (float):

ToEmployeeID: Employee ID for whom the due is made.

IncludeInCalc: True/False

Details:

CreatedByID: ( This is to track which employee id is used as supervisor id)

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description**  **Employee +ve balance : we have to give that much amount (Dr)**  **Employee -ve balance: we have to take that much amount(Cr)** |
| Create  (In Transaction with UpdateAccount) | Amount,  ToEmployeeID,  CreatedByID | 1.Create a Employee Salary in the database  2. Whenever a Salary is created, the account of the other-party () is updated and it’s current balance is added with total dues.  (e.g Current Balance = 0  Now, A salary is created with amount=5000  So new updated balance for that employee should be: 0 + 5000 = 5000 (i.e. we need to pay 5000/-)  )  Update Account (call Employee: UpdateEmpSalaryBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  EmployeeID: Pass “parameter:ToEmployeeID”  CurrentBalance : pass ”parameter:Amount”  Operation=ADD |
| Search | Org ID |  |

## EmployeeSalaryPayment

This table tracks the actual amount given to the employee.

SalaryPayID: Auto Generated ID

TransferDate: (date field) :

Amount: (float):

ToEmployeeID: Employee ID link to employee Table.

FromAccountID: Account ID link to employee Table.

FromEmployeeID: Employee ID link to employee Table.

IncludeInCalc: True/False

Details:

CreatedByID: ( This is to track which employee id is used as supervisor id)

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description**  **Employee +ve balance : we have to give that much amount (Dr)**  **Employee -ve balance: we have to take that much amount(Cr)** |
| Create  (In Transaction with UpdateAccount) | Amount,  FromEmployeeID,  FromAccountID  ToEmployeeID,  CreatedByID | 1.Create a Salary Payment in the database  2. Whenever a Salary Payment is created, the other-party salary due balance is updated and it’s current salary balance is subtracted with total dues.  (e.g Current Balance (due) = 5000  Now, A payment is created with amount=5000  So new updated balance for that employee should be: 5000 - 5000 = 0 )  Update Account (call Employee: UpdateEmpSalaryBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  EmployeeID: Pass “parameter:ToEmployeeID”  CurrentBalance : pass ”parameter:Amount”  Operation=SUBTRACT    3. Whenever a payment is made, the account of the paying-party (Our Employee: From Account) is updated and it’s current balance is subtracted with “Amount” paid.  (e.g Current Employee Balance = 20,000  Now, the employee has made a salary payment of: 5000  So new updated balance for that customer should be:  20,000 - 5000 = 15000)    Update Account (call UpdateAccountBalance Method):  Connection: pass the existing connection object to paticipate in the transaction.  AccountID: Pass “parameter:FromAccount ID”  CurrentBalance : pass ”parameter:Amount”  Operation=SUBTRACT  4. Update ExpenseTracker:updateTotalSalaryPaid(amount, operation)  Where “operation=ADD” |
| Search | Org ID |  |

## Organization

This table tracks the firm/business who are getting registered on the app.

Currently this will be created directly from DB.

Org Id:

Org Name:

GST Number:

PAN Number:

Mobile Number:

Business Category:

Business Description:

Type of Business (Text) : Propertership, LLC, Pvt Ltd, One Person Company, Limited Company

Registered Address:

Company Id: A unique field for a business which is getting registered.

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description** |
| Create |  | 1.Create a Organization in the database.  2. Create a entry into CalculationTracker for this Orgization.  Pass all fields = 0 . |
| Search | Org ID |  |

## CustomerVendorBalanceTracking\_Procedure

This is a stored procedure to track and get following calculation parameters:

**NeedToPayCustomerVendor:**

Sum of all + balances (>0) from Account table of Type=Customer and Org ID=<Replace Org Id>

**NeedToCollectCustomerVendor:** Virtual Column.

(Sum of all -ve balances from Account table of Type=Customer and OrgId=<Replace Org ID>”):

**TotalCustomerVendorBalance** ( Sum (NeedToPay,NeedToCollect) ): Not a physical column. This is virtual column. In MySQL, it is called generated column. (<https://dev.mysql.com/doc/refman/5.7/en/create-table-generated-columns.html>)

|  |
| --- |
| CREATE PROCEDURE `CustomerVendorBalanceTracking\_Procedure`(  IN ParamOrgId INT,  OUT positiveBalance float,  OUT negativeBalance float,  OUT totalBalance float)  BEGIN  SELECT IF(sum(CurrentBalance) IS NULL, 0 , sum(CurrentBalance) ) INTO positiveBalance FROM account WHERE CurrentBalance >0 and OrgId= ParamOrgId and type=’Customer’;  SELECT IF(sum(CurrentBalance) IS NULL, 0 , sum(CurrentBalance) ) INTO negativeBalance FROM account WHERE CurrentBalance <0 and OrgId=ParamOrgId and type=’Customer’;  SELECT IF(sum(CurrentBalance) IS NULL, 0 , sum(CurrentBalance) ) INTO totalBalance FROM account WHERE OrgId=ParamOrgId and type=’Customer’;  END |

A Sample call to the procedure:

|  |
| --- |
| call CustomerVendorBalanceTracking\_Procedure(1,@NeedToPayCustomerVendor,@NeedToCollectCustomerVendor, @TotalCustomerVendorBalance ) ;  select @NeedToPayCustomerVendor,@NeedToCollectCustomerVendor, @TotalCustomerVendorBalance ; |

## CalculationTracker

It is used to keep global calcuation parameters per organization. Whenever a new organization is created, an entry into this table will be created.

Note: Default value of all fields will be 0 in this table.

**Org ID: (Unique):** Organization ID passed from caller.

**UnPaidExpense:** Sum of all unpaid expenses. When a unpaid expenses gets created – Add here.

When an Unpaid expense becomes paid (Update Scenario): Subtract here.

**PaidExpense:** Sum of all paid expenses. When a paid expenses gets created – Add here.

When an Unpaid expense becomes paid (Update Scenario): Add here.

**TotalExpense=** UnpaidExpense+PaidExpense.

Not a physical column. This is virtual column. In MySQL, it is called generated column. (<https://dev.mysql.com/doc/refman/5.7/en/create-table-generated-columns.html>)

**TotalSalaryPaid:** Total Paid Salaries till date.

A= Amount, when a salary payment is created.

TotalSalaryPaid = TotalSalaryPaid + A

|  |  |  |
| --- | --- | --- |
| **Operation Name** | **Mandatory Input** | **Description**  **Employee +ve balance : we have to give that much amount (Dr)**  **Employee -ve balance: we have to take that much amount(Cr)** |
| Create | Amount,  FromEmployeeID,  FromAccountID  ToEmployeeID,  CreatedByID | Create an entry to this table. All fields value defaults to 0. |
|  |  |  |

## EmployeeSalaryDueTotal\_Procedure:

This is a stored procedure to track and get following calculation parameters.

Tracks the total salary to be paid.

|  |
| --- |
| CREATE PROCEDURE `EmployeeSalaryDueTotal\_Procedure`(  IN ParamOrgId INT,  OUT totalSalaryDue float  )  BEGIN  SELECT IF(sum(CurrentSalaryBalance) IS NULL, 0 , sum(CurrentSalaryBalance) ) INTO totalSalaryDue FROM employee WHERE CurrentSalaryBalance>0 and OrgId= ParamOrgId ;  END |

## TotalCashInHand\_Procedure:

Making query on Account Table where Type=Employee and CurrentBalance>0 and Org Id=<Org ID in the row>

|  |
| --- |
| CREATE PROCEDURE `TotalCashInHand\_Procedure`(  IN ParamOrgId INT,  OUT totalCashInHand float  )  BEGIN  SELECT IF(sum(CurrentBalance) IS NULL, 0 , sum(CurrentBalance) ) INTO totalCashInHand FROM account WHERE CurrentBalance>0 and OrgId= ParamOrgId and type=’Employee’ ;  END |

## EmployeeWithBalanceAndSalaryDue\_View:

**Columns:**

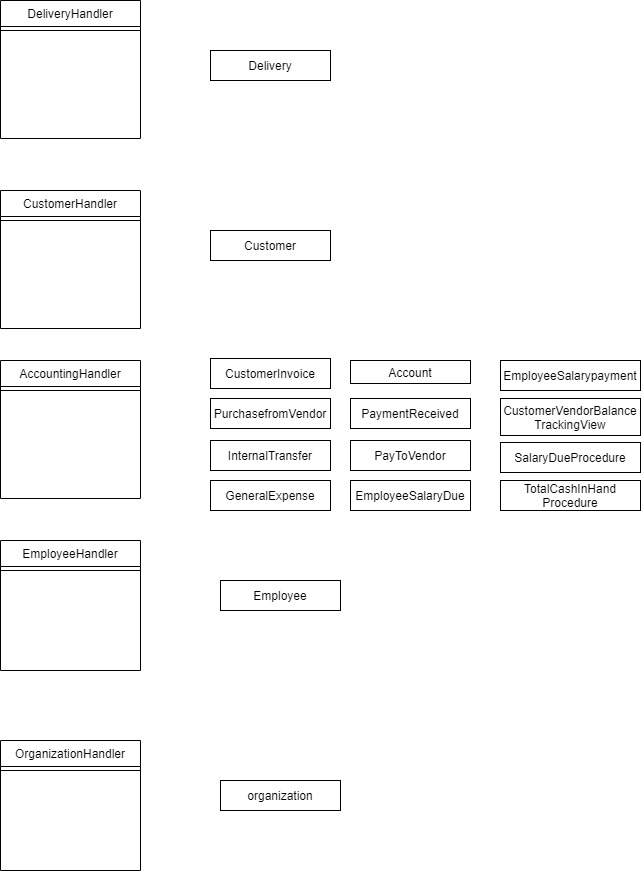
* Employee.Employee ID
* Employee .First Name
* Employee.Last Name
* Employee.Mobile Number
* Employee.loginRequired
* Employee.loginUserName
* Employee.Org Id
* Employee.Type
* Employee .CurrentSalaryBalance
* CalculatedFromJoin\*:TotalBalance

\*To get the “TotalBalance”, a join will be used with Account Table.

|  |
| --- |
| Select employee.\*, SUM(account.currentBalance) from employee, account  Where Employee.Employee ID = Account.EmployeeID  Group By EmployeeID |

**Please note:** For current project requirements, when we are not supporting multiple accounts for an employee, we can simply get the employee’s balance from the “account.currentBalance” field. But in future, we have to modify the view again to support multiple account. So I have decided to right the final query to support the future requirements

# JavaServiceToTableMapping



# Frontend to backend Mapping

|  |  |
| --- | --- |
| **UI** | **Backend** |
| Customer / Vendor | Customer |
| Delivery | Delivery |
| Bill To Customer | CustomerInvoice |
| Vendor Invoice | PurchasefromVendor |
| General Expense | Expense |
| Payment Received | PaymentReceived |
| Vendor Payment | PayToVendor |
| Internal Transfer | InternalTransfer |
| Salary |  |
| Employee | Employee |
|  |  |

# How to get various calculation

A: Positive balance

B: Negative Balance

|  |  |
| --- | --- |
| **Scenario** | **DB Objects** |
| Customer / Vendor Balance (AB)  (Udhari lena + dena ) | CustomerInvoice, PurchaseFromVendor , Payment Received , PayToVendor  **A:** PurchaseFromVendor + Payment Received  500  **B**: CustomerInvoice + PayToVendor  400  Balance: A – B (100 +)  +ve balance : We have +ve balance (We have to give)  -ve Balance: We have -ve balance. (we have to take) |
| Salary Balance (BC)  (Salary dena ) | A: EmployeeSalaryPayment  B : EmployeeSalaryDue  A-B |
| Per Employee Balance (CD)  (apne logo me se kiske paas kitne paise he) | **A:** PaymentReceived, Internal Transfer (ToAccount)  **B:** Expense (From Account) which are paid , PayToVendor, Internal Transfer(From Account), Salary (From Account)  A – B |
| Total Cash In hand  (Sabhi logo jo apne sath kaam kar he unka balance) | + Balance (Per Employee Balance) |
| Total Expense | Sum of expense with Paid=true |
| Due expense (to be paid) | Expense with Paid=false |
| Salary Balance | Salary Paid – Salary Dues |
|  |  |
|  |  |
|  |  |
| Overall (Total Cash in Hand + Customer se lena – Vendor ko dena – unpaid expense – salary dena he (Salary Balance) | Cash In Hand + Udhari lena he – Udhari Dena he |

# Q&A Technical Team

# Q&A User

**Q: What is the purpose of this app?**

This app is solving a very unique problem in businesses. As we know, money is most important for any business so this app gives you or your business, accurate and real-time information about your business health e.g. whether you are making money or losing the money. At any point in time, it gives complete information about your business e.g how much you have to pay vendors, how much you have to take it from customers, how much salary dues, how much is your unpaid and paid expenses thus giving you most updated information of your business. Also, it gives various analytics e.g. your top 5 or top 10 expenses, so that you can look into that and you can reduce them if possible, thus making your business more profitable.

The app/system also maintains various information like customer/vendor and it can automatically remind them about their dues.

One of the strong features of this app is that – it can be used by any simple person who does not know much about Computers/Information Technology. Even if you do not know much business terminology, you can easily use it.

A second great feature is that – it can be used by multiple people working under one business/firm it is designed for that purpose only. In many businesses, more than one person is involved. In India, we have many businesses where father, mother, brothers, friends, and partners are doing the family or any other business. In that case, this is difficult to trace who has how much money, how much to be paid by each person to vendors, who are doing what expenses and if that person is not there how much debt to be taken from customers. This app/system will solve all these day-to-day problems.

**Q: Can this app be used by a retailer to maintain its retail counter?**

**A:** If you want you can do that now but you need to do manual entries for counter sale. We are working on a simple counter sale feature/module on this. But, in a simple scenario, you can take a total to retain sales every day and make an entry into the app/system.

**Q: How to book profit from your business?**

**A:** You should take a salary (create a salary transaction) in the app.

**Q: If someone else is also doing expense for your business e.g your father is paying sometimes when you are not available then how to track those expenses?**

**A:**  There are multiple options for this 1. You can create that person as vendor 2. You can create as an employee 3. Or simply, you can just create a general expense and if you have not paid then use that switch “Paid” in the general expense.

**Q: What can we track as an account(Cr, Dr) in the app?**

**A:**  You can track 1. Customer 2. Vendor 3. Employee

**Q: I want to track my actual accounts like current accounts, saving accounts and my employee’s multiple accounts?**

**A:** Currently, it is not supported but in the application is built to support that and in a future version, we will allow that.

**Q: I am the owner of my business but my business transactions are handled by my account teams, can the app handle the entries from other’s behalf?**

**A:** The app is built to do that but currently it is not supported and in a future version, we will allow that. But, even in that case, you will have to approve the transactions created by others or you can designate some person to that.

**Q: I am the owner of my business and I have two more partners who also would like to see the same level of details as me? Is it supported in the app?**

**A:** Yes. All the functions built in the app are configurable. So you can control what rights you want to assign.

**Q: Does Salary paid to the employee come under the balance for a person?**

**A:** No. Salary is personal or profits to that employee. That is not counted for company balance.

**Q: I can enter initial balance for customer/vendor and individual employee but how can I enter initial balance for employee salary?**

**A:** You can create a Salary Due after creating the employee. That will work as an initial balance.

**Q: What if I want to have an initial balance of unpaid expenses I have?**

**A:** You can go to settings and update the “initial unpaid expense” setting value.

**Q: How do I handle material return case meaning I sent some material of 20,000Rs and he returned Rs. 5000/- material, he just took/bought only 15,000/- material but I have already created 20,000Rs entry into the app?**

A: You can create a bill entry of -5000 ( minus 5000 Rs.) that will adjust your actual sale of 15000 (20,000 – 5000 = 15000Rs)

**Q: I have to create entered a wrong amount in Expense or Salary Due or Payment Received etc, but I have seen that the app/system is not allowing me to edit the amount.**

**A:** Yes, as this app/system is of financial category, we cannot allow the amounts to be edited as someone can edit the amount and can do fraud. But, you can do a reverse entry (a corresponding negative entry to achieve the same result. An example is given in the above scenario.

**Q: I want to track my material quantities and how much I have sold, how can I do that?**

**A:** The system/app is not designed for that purpose currently, it is designed to manage the money which is circulating across your organizations and provide you, accurate business health at any point in time.

**Q: One of my vendors has created a due against me and I can log in with my mobile and OTP but I can see what the vendor has created for me. I cannot do anything.**

**A: Y**ou are not registered currently on the app/system. The app allows you to use the basic function without registration. If you are running a business or a business owner or a caretaker of a business then you can register your business. Once you have registered your business then you will get a “Company ID” and password. When you are login with “Mobile Number”, “Company Id” and “Password” then you will be able to see all the features.

**Q: I am a business owner of more than one business e.g 5 business, do I need to register with a 5 different number?**

A: No, you do not need to register with a different number. When you are registering a new business then the system will provide you unique Company ID. You need to log in with respective Company Id to manage that specific business. All different businesses under you will still be different for the system. You can see the balance of different business together at this time.

**Q: If I see my business dashboard then I do not see that total amount due for employees accounts (negative balance of employee) are used while doing overall calculation of my business health. Is it by mistake?**

**A:** NO, it is not calculated because the account due of employee may happen due to one of the reasons 1. Vendor / Customer Payment. 2. Expense payment 3. Salary payment. All of these are already included. If we consider the due balance of each employee into the calculation then it will be a duplicate and will be incorrect. That is why, we have not included that into the calculations.

# Initial Onboarding of an Organization

Create an organization.

Create Customer and Vendor – Update their initial balance on the screen. [Need to collect, Need to Pay]

Create Employee and update initial balance there on the screen. [CASH IN HAND]

Settings: Update the initial unpaid expense. [Unpaid Expenses]

Salary: Create Salary due for each of the employee. [Salary Due]

# System Processes Explanation

|  |  |  |
| --- | --- | --- |
| **Scenario/Problem** | **Process** | **Action Item** |
| How my customer/vendor can see entries created for him/her? | The person has to download the app and he can simply put his registered mobile (that you have entered for him, make sure it is not incorrect) and the user will receive OTP there and he will be able to log in. However, this will only allow a basic view which is just to see entries created for him by any vendor/customer. | Develop this feature |
| If a business owner has more than one business, how does the owner register his next firm using the same registered mobile that he/she used earlier? | That is possible. He can register for the next business with a registered mobile number. The system will alert that you are already registered a few businesses earlier so make sure that you are not duplicating. After a unique business verification, the system will allow. | 1.Unique business registration: we need to develop some ways to uniquely identify a business. |
| How can we increase the downloads of this app? | Let's say one of an organization is using our app, once they are entering their vendors/customers, we can send a notification to the vendor/customer also to download the app. | Develop this feature |
| What about a user who is entered by any organization/business as its’ employee and made “login not required”. Now, let’s say, if that user is also running his/her own business and want to use our app to manage his/her business, will the user be able to use our app? | Yes. The user can still register his business where he/she will receive a unique company id.  The user will see screen/function in the app, as per the company id he is using to log into. In this case, if he is using the company id where is he the employee and not able to log in then, he will not be able to log in. If he is using his own registered business’s company Id then he will be able to log in and manage. |  |
| What will happen in case of Loan Taken (money interered from outside ? |  |  |
| What if if a give loan to someone ? |  |  |

# Direct Data Load in MySQL ( No need to maintain from code)

**Expense Category:**

* Machine Parts Purchase
* Machine Maintenance
* Transporting / Courier Charges
* Non-Company Related
* Deposit
* Vendor Payment
* Misc
* Marketing

**List of Citites**

* Barufatak
* Dhamnod
* Thikri
* Khargone
* Badwani
* Rajpur
* Indore
* Rau
* Pithampur
* Mandleshwar
* Bistan
* Lohari

**User Roles**

* Production Manager
* Operations Manager
* Accounting
* Sales Lead
* Admin

**Function**

* *Create Employee*
* Update Employee
* Create Customer
* Update Customer
* Delete Customer
* Create Expense
* Update Expense

**Nitesh Finding:**

**Customer**

1. Lets make firm name mandatory. If end-user is not providing firm name then we will copy customer name to firm name (from backend)

2. Email field to the customer will be added later.

3. Whatsapp reminder will be integrated later. (Mobile2) can be used for storing the whatsapp number.

4. For Mobile Number of a customer, address book integration can be done.

**Expense will cover all type of expense like**

* Raw Material Purchase (Panni, Polythene, Gum, Silver, Paper, Core pipe, Die, Machine)
* Machine parts
* Machine maintenance
* Transporting

For the first version of the product: We will create all outgoing as expense and ToPartyName as free text field.

**Money management**

|  |  |
| --- | --- |
| **Incoming** | **Outgoing** |
| * Payments from Customer * Return from vendor * Scrap Sale | * Expense * Inter person transfer * Vendor Payments |

**Scenario:**

* Scrap Sale : (They are not customer, but they can be managed as customer)
* Die Purchased from Rizvi and then we made payment. Then we returned die to Rizvi and he is making a payment:

**Solution:** Create Rizvi as customer and make a bill against him and payments.

Functions Available:

**Customer Balance Tracking**

You can go to customer view and see the balance. + Balance refers that they have deposite of that amount with you. A negative balance refers that the customer has to pay that much of amount to you.

**Your Balance:**

Customer balance can be calculated from Billing and Payment Received.

Individual balance can be calculated from Payment Received