**SUSTAINABLE OUTREACH AND UNIVERSAL LEADERSHIP LIMITED  
(SOUL)**

****

**KIMS HOSPITAL - INVOICE TRACKING SYSTEM**

**DESIGN DOCUMENT**

1. **Department:**
   1. **Appending Company name With Department after Creating:**

The definition ‘autoname’ will be used to generate the primary key / name of the department depending on the inputs given by the user. For example, if the input on the department\_name is “Consulting” and the name of the underlying company is ‘SOUL’, the department’s name will be set to “Consulting - SOUL”.

Steps to be executed:

**Step 1:** Import get\_root\_of function from frappe.utils.nestedset, //get\_root\_of is a method for getting root element of a DocType with a tree structure

**Step 2:** User inputs department name, Department Abbr, company, Is company and Approves.

**Step 3:** Find root node of the department of the current department by passing document\_name in get\_root\_of function

**Step 4:** check if department name is not same as root name

**Step 5:** IF root name and department name in the form is not equal to root

Set name of the document to an appended string of department\_name and company

ELSE

Department name set by the user will be set to Document name

**Step 6:** Save.

**1.2 Assigning new Departments with Parent Department:**

Assigning Parent Department to each new department that is added.

Steps to be executed:

**Step 1:** Under validate controller we will check if the parent\_department checkbox is ticked or not.

IF true

Set current Department as root department

ELSE

The current department will be under all departments.

**1.3 Setting Primary Key for Department:**

The definition ‘before\_rename’ will get the value of the company abbreviation mentioned by the user and pass it onto the autoname definition to enable the construction of the primary key of the record entered.

Steps to be executed:

**Step 1:** Pass department name and company name entered by the user as a parameter in get\_abbreviated\_name function which appends department\_name and company\_name to return the new department name as Primary key for the record that is created.

**1.4 Deleting Documents:**

The ‘on\_trash’ definition will ensure that the department is deleted properly from the database as well as the tree view.

Steps to be executed:

**Step 1:** Define delete\_events method having parameter ref\_type and ref\_name

**Step 2:** Get list of DISTINCT events from tabEvent and tabEvent Participants where tabEvent name matches with TabParticipants parent field and tabEvent Participants ref\_type is the doctype name, ref\_name should be document name.

**Step 3:** IF events is not null run a frappe.delete\_doc for the event found.

**Step 4:**Pass department name and doctype name to delete\_events method

**Step 5:** Save

**1.5 Getting Child for a Parent Department**

The ‘get\_children’ definition is called by the department\_tree.js to get all the records of all the departments which fall under a parent department, another parent department falling under the previous parent department which also has children and so on.

Steps to be executed:

**Step 1:** Store document name as value, is\_group as expandable in a variable.

**Step 2:** Set filters which should be applied.

**Step 3:** IF company is equal to parent:

Apply filter by name of document by getting root value of department

ELSEIF only company field is present:

Apply filter based on parent as parent\_department and company.

ELSE:

Apply filter based on parent as parent\_department

**Step 4:** return all records of doctype based on fields and filters that are applied and order by name of doctype.

**1.6 Adding new department as node in Department tree view**

The ‘add\_node’ definition imports the treeview from frappe.desk.treeview and the function called make\_tree\_args takes the arguments for the treeview and forms a tree in the frontend. Whenever a new record is created, the tree is updated accordingly.

Steps to be executed:

**Step 1:** Import make\_tree\_args from frappe.desk.treeview

**Step 2:** Pass form data as dict in args

**Step 3:** Call make\_tree\_args by passing args to create treeview for department doctype

**Step 4:** IF args.parent\_department is equal to args.company

Set args.parent\_department to None

**Step 5:** Insert args by passing in frappe.get\_doc method.

**Step 6:** Save

1. **Employee:**

**2.1 Creating user automatically after creating Employee:**

When we Submit the Employee form we separately need to create user so that the employee can login and use the system. The purpose here is to create User with as soon as a employee is created.

Steps to be executed:

**Step 1:** Check if the Email entered is not a user by checking his email that is entered.

**Step 2:** Create a frappe.get\_doc instance by passing docytpe as user, first\_name, last\_name, email, username, role\_profile\_name and user\_type.

**Step 3:** Save the instance

**Step 4:** db.set fot the instance that we have created to set the user email field in employee doctype.

**2.2 Creating a checkbox to disable the user**

A checkbox needs to be created so that the user can only login and use the system if it is checked else they will not be able to use the system.

Steps to be executed:

**Step 1:** Fetch all enabled users from database

**Step 2:** IF User\_list is not Null

IF Enabled value in document is 0 and enabled value in user table is 1

2.2.1 Get all users and email who have enable as 1 in user table

2.2.2 Get User ID of all users with enable value as 1.

2.2.3 update enable value as 0 in user table

2.2.4 Save the document

IF Enabled value in document is 1 and enabled value in user table is 0

2.2.1 Get all users and email who have enable as 1 in user table

2.2.2 Get User ID of all users with enable value as 0.

2.2.3 update enable value as 1 in user table

2.2.4 Save the document

**2.3 Updating Users role profile from Employee form**

If a users designation changes it should be changed in the system too. By default this is done in User doctype but this should be updated from the Employee doctype.

Steps to be executed:

**Step 1:** Fetch all enabled users from database

**Step 2:** IF User\_list is not Null

IF Role profile is Null

Update the role profile with selected role profile.

**2.4 Changing User ID from Employee Table**

User id is set when a user is created and if we want to change it it must be set from User doctype, This method allows us to change users email from employee doctype.

Steps to be executed:

**Step 1:** Fetch all enabled users from database

**Step 2:** IF employee data is not null

IF email field in employee is diferent from employee field in user

2.4.1 Update old user ID (email) is replaced by new email.

2.4.2 Rename User doc

2.4.3 Commit the changes

2.4.4 Get email field from user table

2.4.5 Update email in email field with new email id.

2.4.6 Save the user doctype.

2.4.7 Set the updated user name in employee table

**2.5 Changing password from User Table**

To change password for any employee it is changed from the user doctype, This method allows us to change users password from employee doctype.

Steps to be executed:

**Step 1:** IF new password field in employee doctype is not null.

IF user field is not null

2.5.1 Get email of employee from user doctype

2.5.2 Update new password with old.

2.5.3 Save the document

2.5.4 Set new password field in employee table as blank

ELSE

Set new password field in employee table as blank

1. **PO Consumable Invoice:**

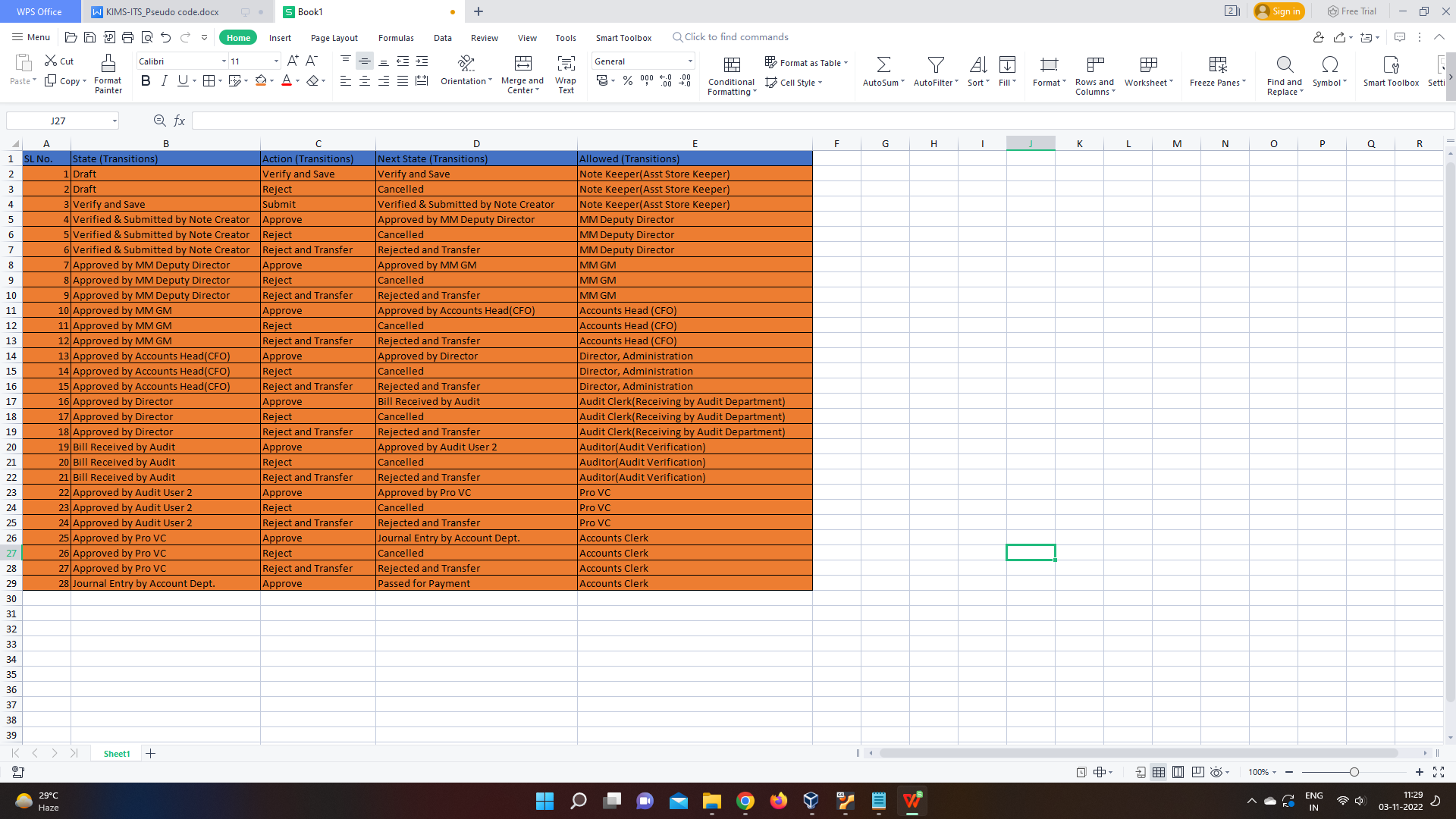
PO Consumable invoice is a doctype created to enter Note sheet related to consumable Invoices.

* 1. **Workflow design from Frontend:**

A total of 14 Workflow states will be used while triggering workflow for this doctype which includes:

1. Draft
2. Verify and Save
3. Verified & Submitted by Note Creator
4. Approved by MM Deputy Director
5. Cancelled
6. Approved by MM GM
7. Rejected and Transfer
8. Approved by Accounts Head(CFO)
9. Approved by Director
10. Bill Received by Audit
11. Approved by Audit User 2
12. Approved by Pro VC
13. Journal Entry by Account Dept.
14. Passed for Payment

The Transition states for PO Consumable workflow consist of 28 Transition rules which are mentioned below:



* 1. **Childtable Total calculation:**

There will be a child table for Entering Invoice, PO and all other attachments also the amount of the Invoices for which the Total would be calculated and si[layed in the parent form. The steps to be excecuted for total excecution are:

**Steps 1:** Store the value of each row of childdoctype in a variable.

**Step 2:** Define a variables for storing total and initialize it to 0.

**Step 3:** ForEach value in childdoctype,

Add total=total+Each value of childdoctype

**Step 4:** Set total with the value calculated and reresh the field.

* 1. **Deducting TDS and Advance payment from childdoctype:**

The Grand total of the invoice would be calculated after deducting TDS and advance payment if paid any. The Steps to be excecuted are:

**Step 1:** Declare a variable and set it to 0 to calculate the new total amount.

**Step 2:** ForEach value in childdoctype,

Grand\_total = total - tds - advance

**Step 3:** Set grand\_total with the value calculated and reresh the field.

* 1. **Appending Authorized Personal details in child doctype and Giving Users option to trnasfer to Invoice other users in workflow:**

While Workflow for a doctype is active it will be passed from a number of users those who will Approve/Reject it. We will Identiffy the User accessing the Invoice from Session variable. First we will append the Authorized childdoctype with the user information Accessing the Invoice

The Steps to be excecuted are:

**Step 1:** Get session user email from frappe.session user

**Step 2:** IF Worklow State is not equal to “Cancelled” and “Reject and Transfer

IF session\_user not NULL

Get employee details like name, salutation, designation, department

IF employee details found:

Set Flag to Yes

Create a variable(Obj) to store signature authority details

FOR each access to Invoice append authorized childdoctype with the details

IF variable(Obj) is not NULL:

IF approval\_status in childdoctype is equal to workflow\_state:

Set Flag to No

Update approval\_status field in childdoctype with current workflow\_state

Get previous workflow\_state from database

IF previous\_state not found

Set previous\_state to NULL

ELSE

Set Current workflow\_state to previous\_state

//Updating date of receivable

Assign the users input of approval date to date of approval

IF date\_of\_receivable is NULL

Set today's date as date\_of\_receivable

IF flag is YES

Append authorized\_signature table with Employee\_id, name, designation, approval date, receivable date, department, approval state, previous state

IF flag is No

//file is modified without changing workflow\_state

FOR all entry of authorized\_signature table

Set previous state to current state

Update user data who is doing the changes

ELSE

Throw(Employee not found)

ELSE

//Transfer function for transferring workflow

IF Workflow\_state is equal to Reject and Transfer

Create a count variable and initialize to 0.

FOR values in table authorized\_signature

IF transfer\_to variable is active and disapproved\_check is 0

Store previous status and name of invoice

IF name is not NULL

Get employee data from session

IF employee data present

FOR all values in authorize\_signature table

IF previous name and current name of employee is same

SET all previous authorization to disapprove

Commit the changes

Update workflow\_state to previous workflow\_state

Commit changes

ELSE

Throw(Employee not found)

ELIF Count is greater than 1

Throw(Transfer to employee not selected)

* 1. **Email requirement:**
* Mail should be send to the supplier when user receivers the invoice.

Set on Condition, Workflow\_state == “Verified and Submitted by Note Creator”

* 10 days before the thrash hold payment date if payment is not cleared to all the user of the workflow.

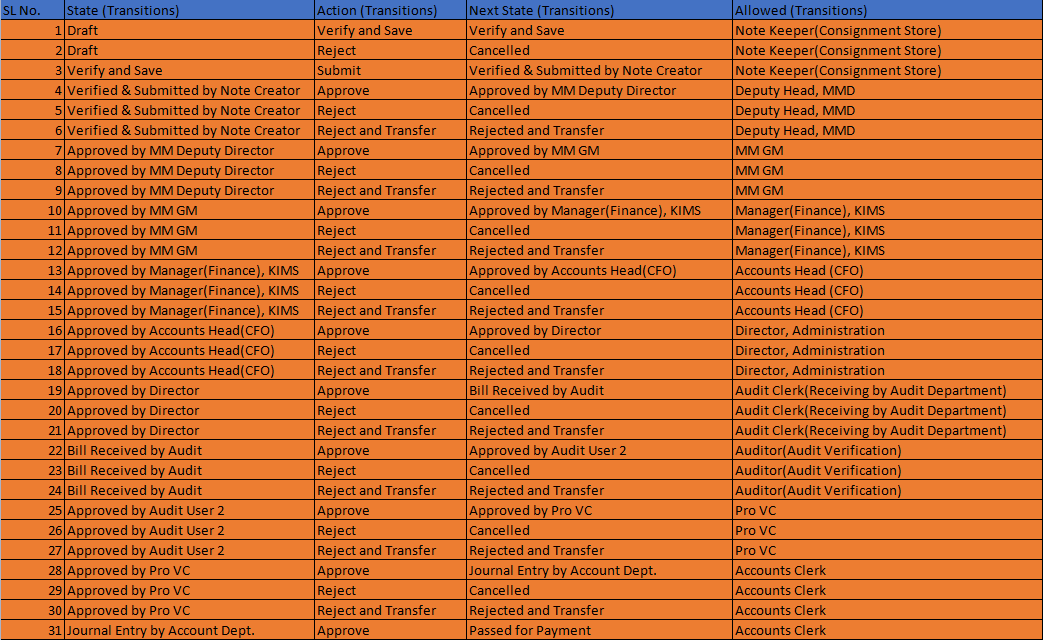
In Notification, Send Alert on -> Reference Date -> 20 days after creation date

* When payment is processed and payment is done supplied will receive mail of confirmation of payment.

Send on condition, IF Workflow\_state ==Payment Done

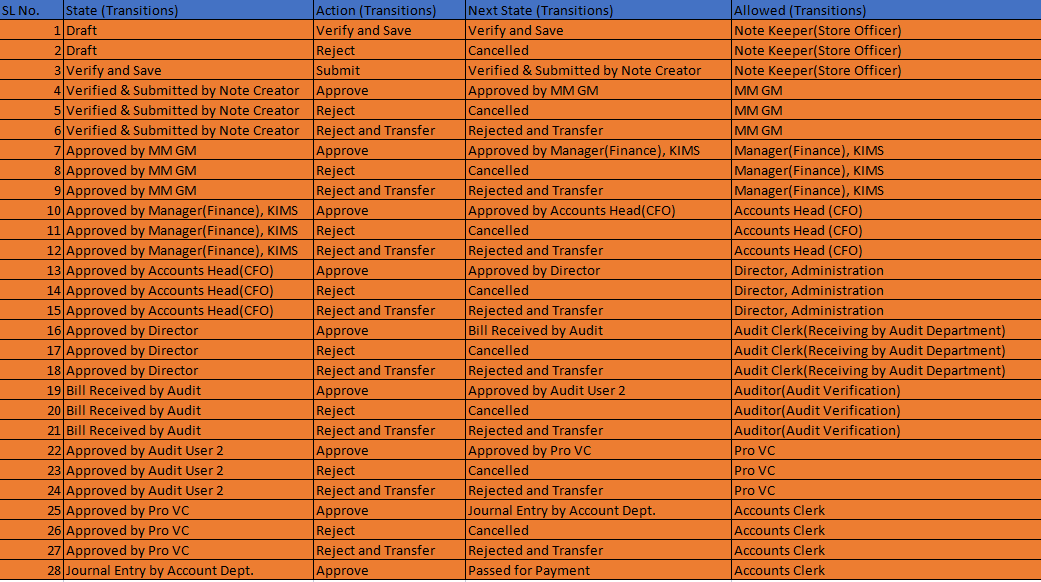
1. **PO Consignment:**

The Transition states for PO Consignment workflow consist of the following Transition rules:



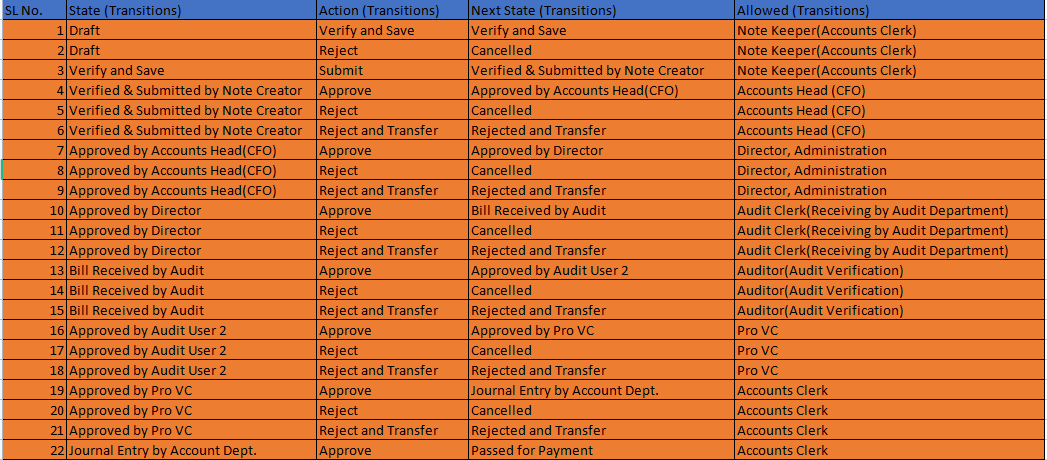
1. **PO Equipment:**

The Transition states for PO Equipment workflow consist of the following Transition rules:



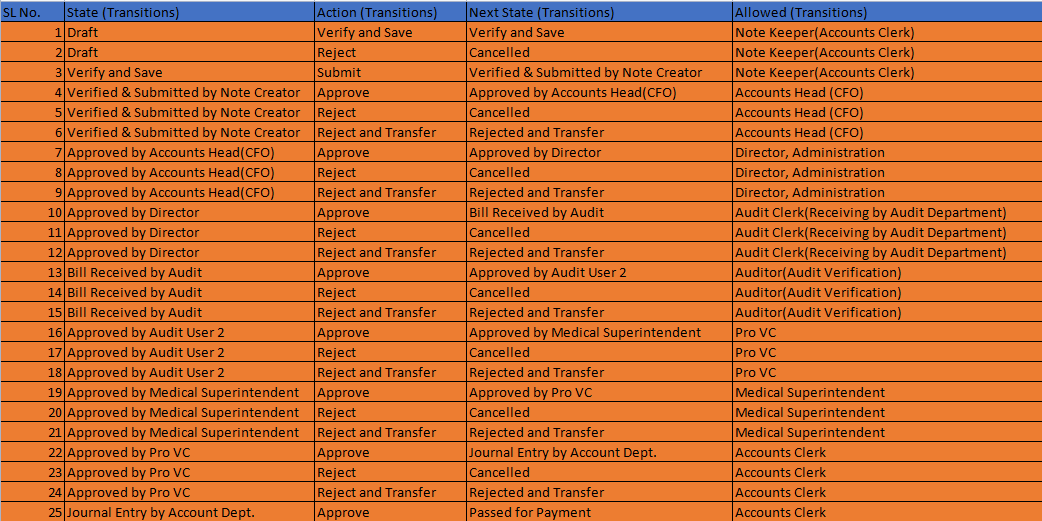
1. **PO Contract:**

The Transition states for PO Contract workflow consist of the following Transition rules:



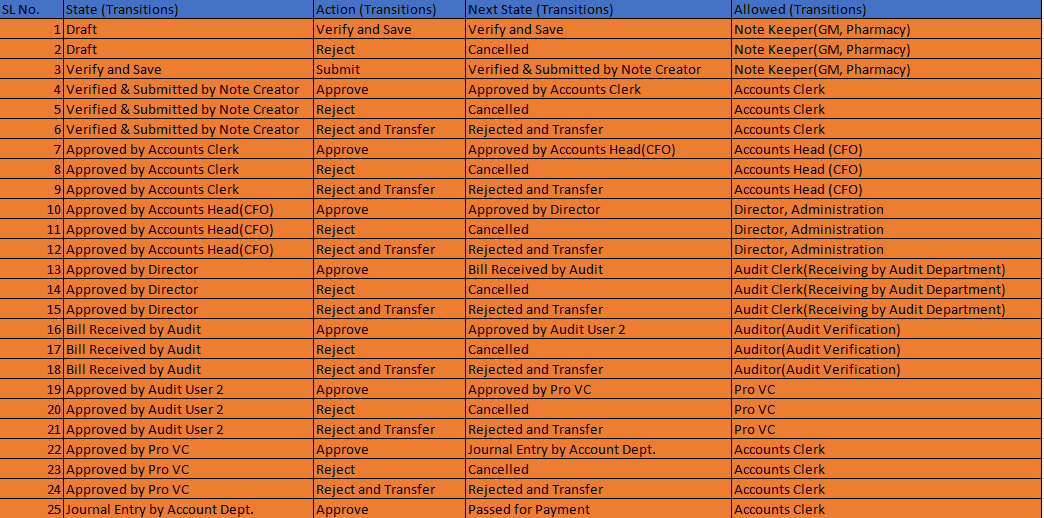
1. **Non PO non-Contract:**

The Transition states for Non PO non-Contract workflow consist of the following Transition rules:



1. **Pharmacy:**

The Transition states for Pharmacy workflow consist of the following Transition rules:



1. **Patient refund:**

The Transition states for Patient refund workflow consist of the following Transition rules:

**9.1 Workflow Transitions:**



**9.2 Patient refund:**

Total Bill = Insurance - Approval of TPA/Insurance/Corporate/OSTF - Non-admissible item/Discount amount

= X1

Amount Deposited by patient

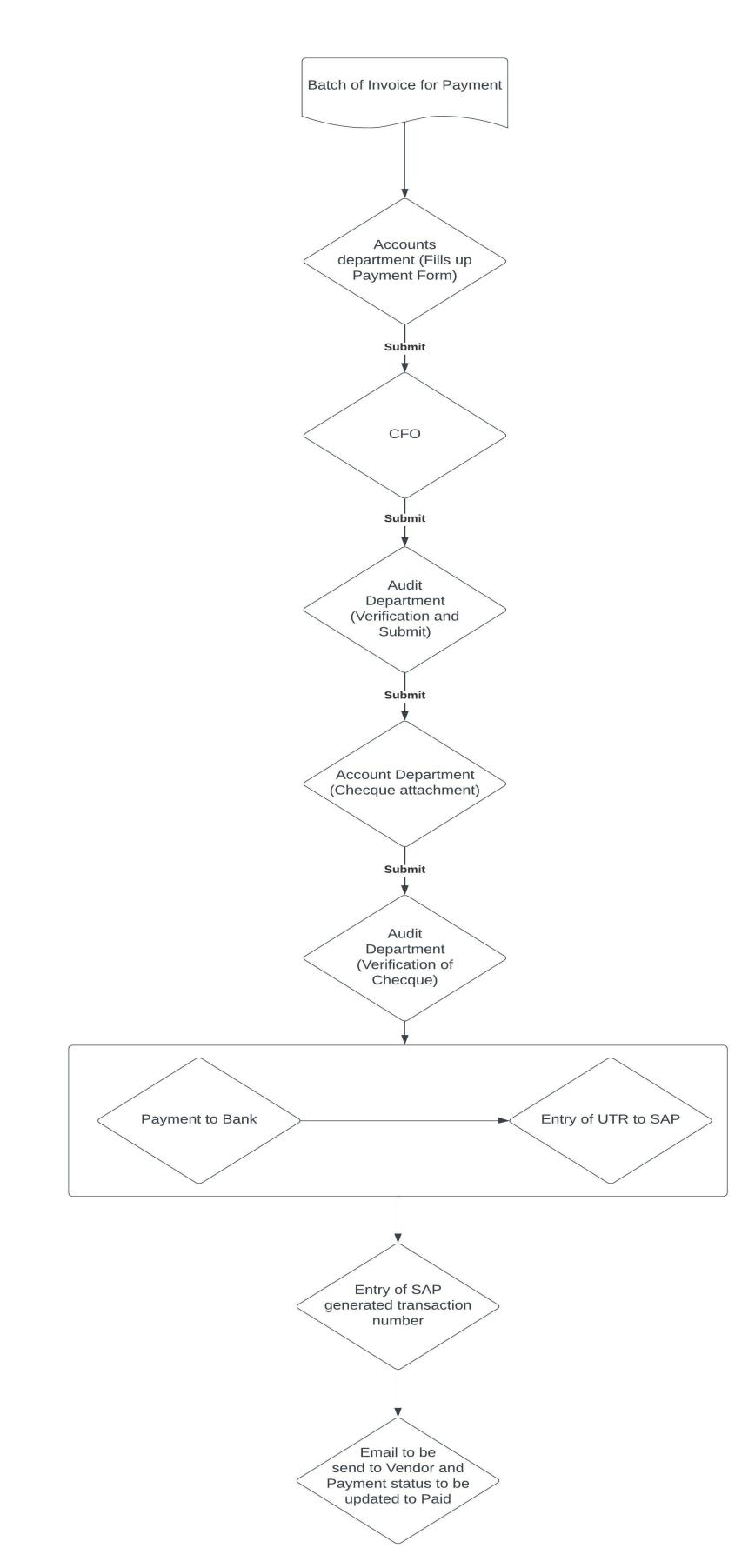
- Cash Refund

= X2

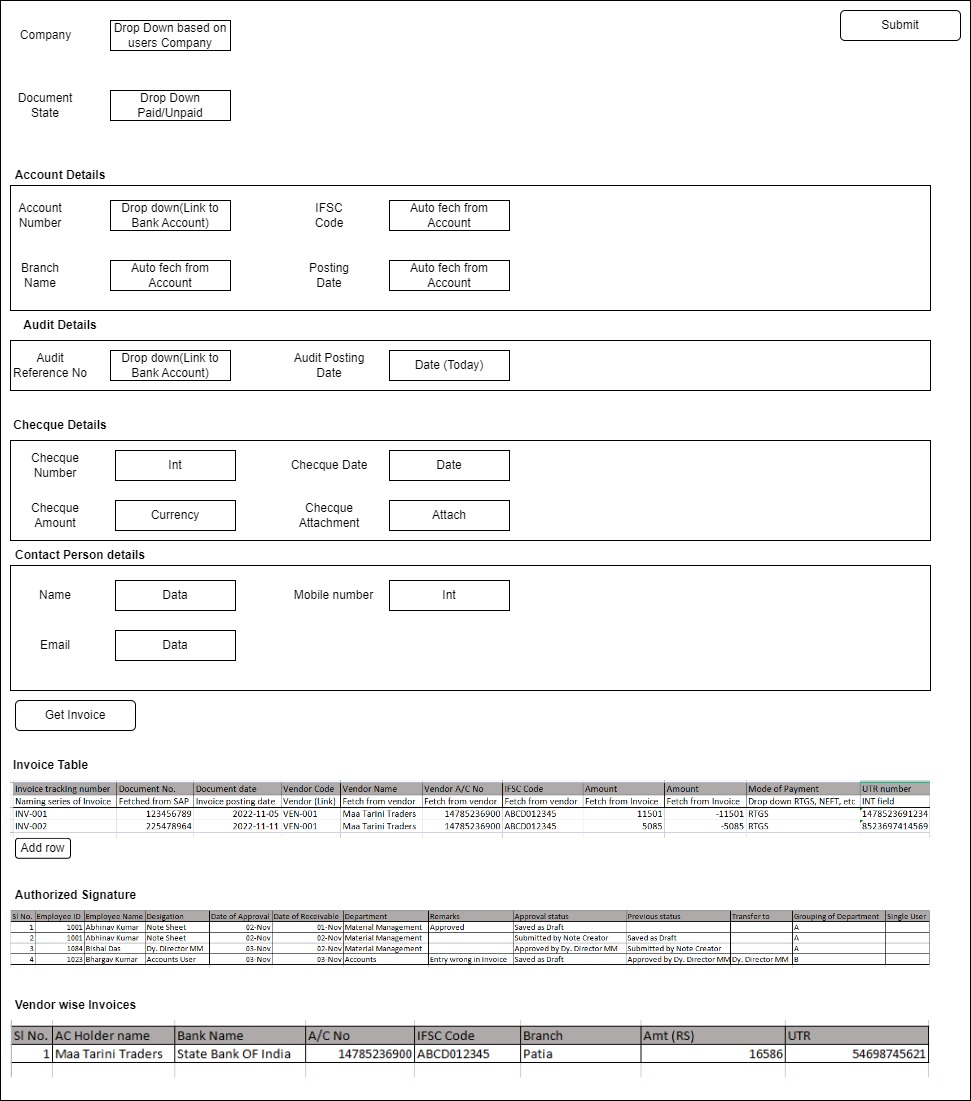
Net Refund = X2 - X1

1. **Payment Flow:**

This workflow will be triggered for all Invoice types after they are in unpaid state and Passed for payment.

****

**10.1 Payment Flow overview**



**10.2 Filters will be applied based on**

1. Posting Date (Start date, End date)

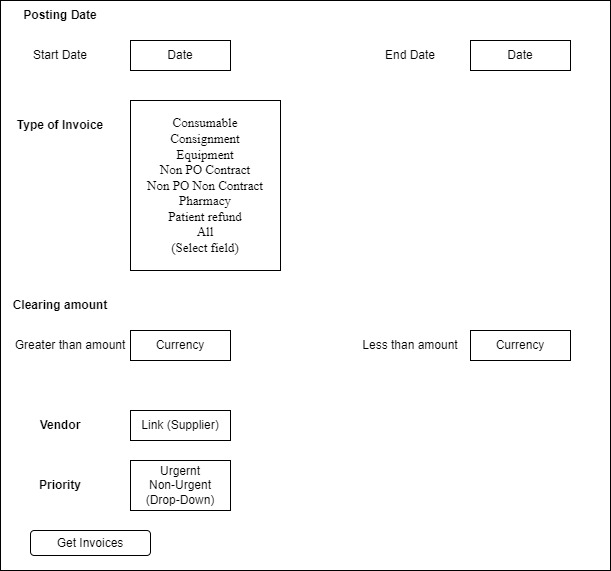
2. Type of Invoice (Consumable, Consignment, Equipment, Non PO Contract, Non PO Non Contract, Pharmacy, Patient refund)

3. Clearing Amount (Greater than amount, Less than amount)

4. Vendor wise Invoice

5. Priority

6. Only Invoice with Workflow\_status Process for Payment should be fetched when After selecting appropriate filter necessary result will be populated in the invoice child table.



**10.3 Form Logic:**

**Step 1:** Document will have fields(Company, Document Status, Account Post Date, Paying Bank, Bank A/C No, Branch Details, Bank Name, IFSC Code, Audit Reference No, Audit Posting Date, Cheque No, Cheque Date, Amount, Cheque Attachment, Note Sheet Attachment, Contact Person, Contact Phone No, Contact Email Id)

**Step 2:** A **Get Invoice** button will be created to Fetch all the Invoices with payment status **Unpaid** by filtering based on the condition that is set in the popup.

\* fields for POP up (Posting Date (Start Date, End Date), Type of Invoice, Clearing Amount (Greater than amount, Less than amount), Vendor, Priority)

**Step 3:** 3 Childdoctypes will be created:

2.1 For Storing Invoices (All entries)

2.2 For Appending Signatures

2.3 For Grouping Vendor wise Amount(eg, A Vendor has 3 invoice entries while we click Get Invoice button but the payment for this vendor will be done 1 time so all the invoices will be grouped to get total amount that will be paid to the vendor)

**Workflow State 1:** Vendor information and SAP transaction number will be fetched/entered as needed and submitted by Accounts User.

**Workflow State 2:** Goes to CFO for approval.

**Workflow State 3:** Goes to Audit to be reviewed where a payment sheet(a specific print format) is prepared from our Application.

\* Vendor wise grouping for payment will be done at this stage which will added to a new childtable.

**Workflow State 4:** Goes to Signing authority for approval.

**Workflow State 5:** After verification Accounts department Attach the cheque for payment which they prepared also enters Its details.

**Process outside our Application:**

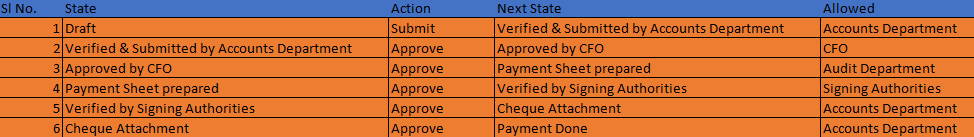
1. Goes to bank from where UTR number for payment is generated(Manual process)

2. UTR entered to SAP to get SAP generated Transaction ID (In SAP)

**Workflow State 6:** Accounts department enters SAP generated Transaction ID and UTR in payment sheet.

**Workflow State 7:** Email will be triggered for Payment containing the transaction number to all vendors and Unpaid status will be changed to paid in all notesheet.

**10.4 Stages of Workflow:**



1. **Dynamic Workflow Tool:**

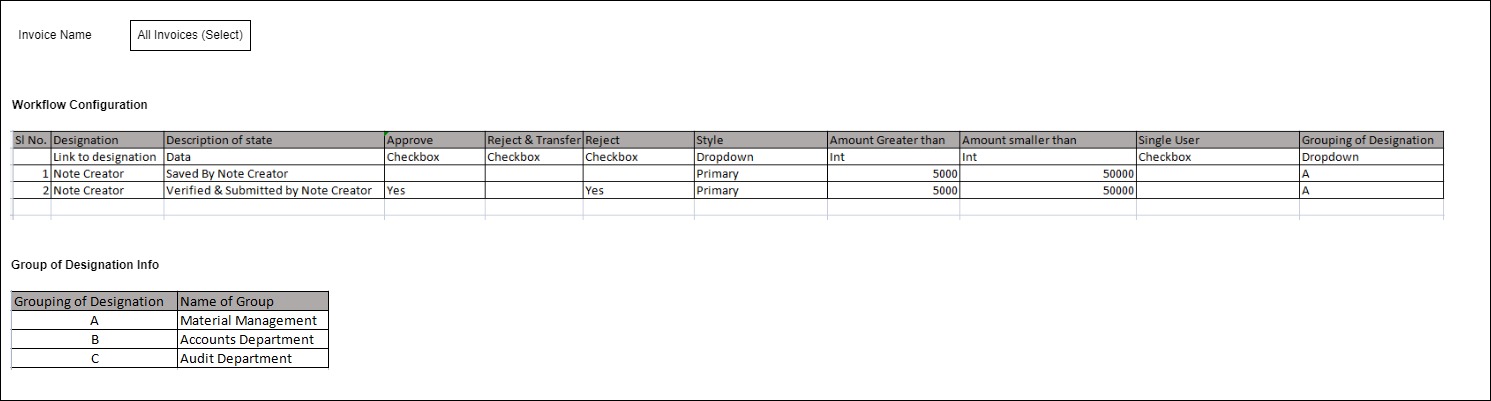
This tool is useful for Configuration of Workflow from a user level, to modify state of workflow with out changing from the workflow doctype.

**Input:**

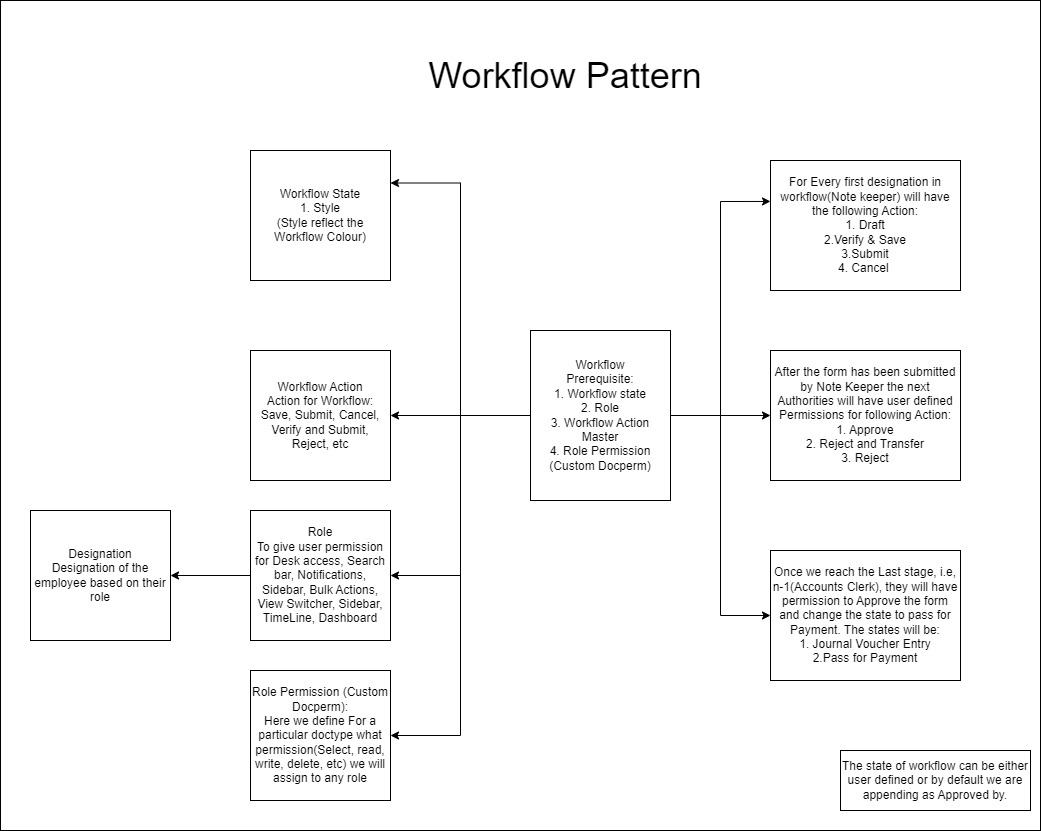
* Doctype name(to be selected from link field)

**Child doctype Values for each line item:**

* Designation(role) to be used in Workflow
* Workflow State name (description)
* Approve, Reject, Reject & Transfer (checkbox)
* Style (colour)
* Amount wise Condition for Execution of Workflow
  + Greater than
  + Less than amount
* Permission Level-
  + Inclusive
  + Non-Inclusive
* Single Value (Checkbox)
* Group by (Select)



**Workflow Prerequisites:**

****

**Programming Logic:**

**Step 1:** User Selects the Doctype for which they want to modify Workflow.

**Step 2:** User add all the States of the workflow in a child doctype as per sequence as they want it to be changed.

**Step 3:** User adds Description of workflow state, who can access the form during that State, their permission to Approve, reject, reject and transfer, Style, Financial criteria to be considered for execution of that state.

**Step 4:** For Excecution of Financial Criteria, select Permission Level as Inclusive or Non-Inclusive, In Inclusive the state will be included in workflow when the criteria is fulfilled. While In Non-Inclusive the criteria will not be included when the criteria is fulfilled.

**Step 5:** Click Single-User Checkbox if the user of the state is a single user

**Step 6:** Select Level of Groups for the designation of the user that would be group together in one department.

And saves the form.

**Step 7:** After saving this form a **new workspace is created** with **modifications** for the doctype that user has selected based on the input in the form. As we can have only one active form for every doctype so the previous form is set to **Inactive** while the newly created form becomes **Active**.

**Step 8:** In the backend the following functions must be created:

Create\_designation (If role not previously present)

Create \_RolePermission (Permission to doctype for newly created role will be given)

Create\_WorkflowState (Worklow state would be created)

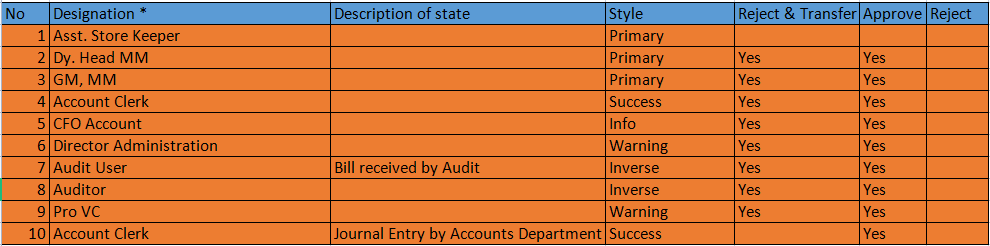
Create\_WorkflowAction (Label at each workflow state)

Create Workflow (Create the new workflow)

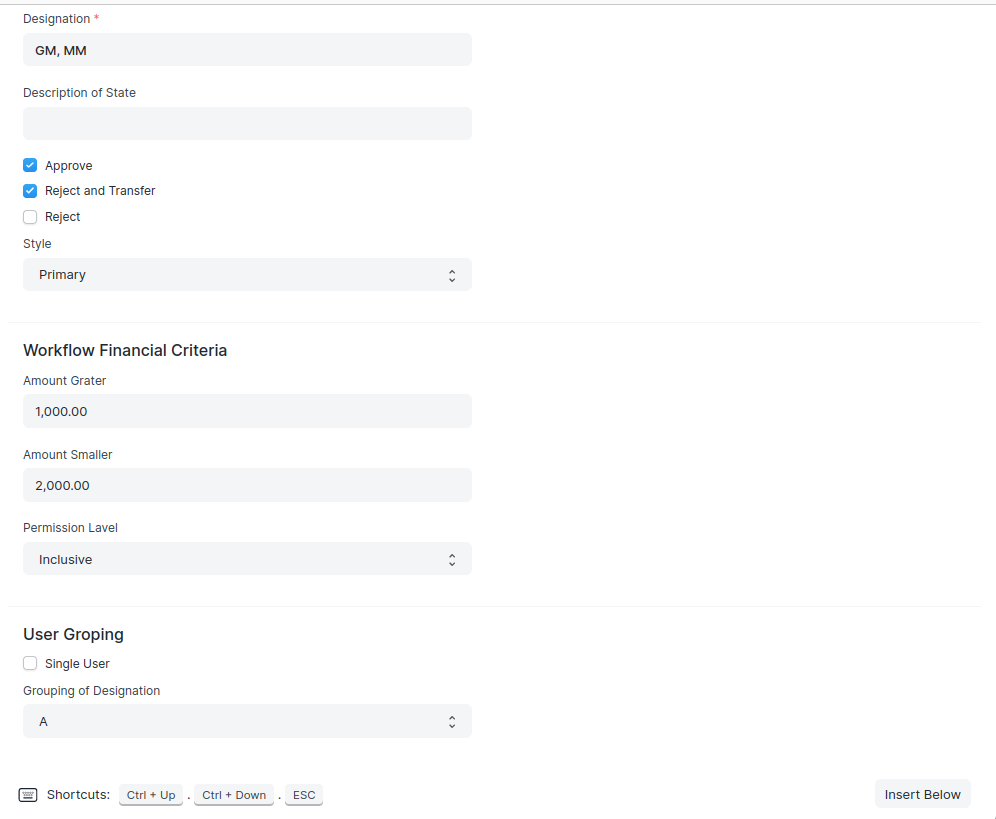
**Output:**

A new Workflow for the selected doctype.

**Workflow Configuration:**



Fields for Each line item



* 1. **Sub processes:**
     1. **Creating Designation Doctype:**

Designation will be the role that will be set to workflow who will be accessing it in a particular state.

Steps to create Designation:

Step 1: Get all Roles through frappe.get\_all

Step 2: IF there are no results, Creates a new instance and sets its role name to "Role".

Step 3: Set the designation entered by User as role.

Step 4: Set role\_name=0, search\_bar=0, notifications=1, list\_sidebar=0, bulk\_actions=0, view\_switcher=0, form\_sidebar=1, timeline=1, dashboard=1.

Step 5: Save the data.

**11.1.2 Creating Role permission**

The role permission to access the doctype for new role needs to be created before they can access the doctype. To do that we create Custom docperm.

Steps to create Custom Docperm are:

Step 1: Get all values for Custom Docperm by Filtering the result using the designation and result used in current workflow.

Step 2: If role permission not found

Create a new instance for Custom docperm, setting slected Invoice name as document name, designation as role.

PermLevel=0, select=1, read=1, write=1, create=1, delete=0, submit=0, cancel=0, amend=0, report=1, export=1, share=0, print=1, email=0

Step 3: Save the document

Step 4: Update import value to 1 from the table `tabCustom DocPerm` where document name matches with current doc name.

**11.1.3 Creating workflow state**

Workflow state are the state of the document when a workflow is active for that particular form.

Steps to create workspace state includes :

Step 1: Iterate a FOR loop in child table to get all existing values.

Step 2: IF idx(Child doctype element number) == 1:

Assign a list(list\_info) with values Draft,Verify and Save,Verified & Submitted by Note Creator, Rejected and Transfer,Cancelled.

Step 3: FOR z in list\_info

Get all workflow state filtered by the value in list\_info and return name of state, style

Step 4: If state not present create a new state using frappe.new\_doc

Step 5: If description field is empty assign “Approved by” with designation for state

Step 6: Set State as Pass for payment if no other state present in list.

**11.1.4 Creating Workflow Action master**

Workflow action master is the label of button that will appear while the transition of workflow.

Steps to create workspace Action includes :

Step 1: Store all action values in a list

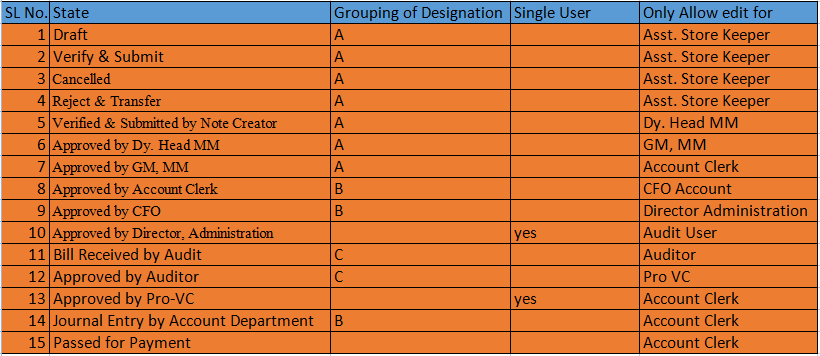
Step 2: Iterate through all values of childtable, to get all values For Workflow

Step 2: If not present, add using frappe\_new\_doc.

**11.1.5 Creating Workflow**

Creating workflow through backend while user only enters on which doctype they want to apply this workflow.

Workflow States:



Workflow Transition:



Steps to create workspace includes :

**Step 1:** Write a frappe get\_all query to get all values for workflow filtered by document type(Invoice name)

**Step 2:** Set name of workflow by appending Invoice name with length of all existing workspace+1.

Eg, If a new workflow created for PO\_Consumable, Name will be:

PO\_Consumable\_1

**Step 3:** Create a new instance for Workflow with values:

Name of workflow=document name (from step 2)

Doctype\_name=Invoice name

Is\_active=1

Override\_status=0

Send\_email\_alert=1

Workflow\_state\_field=workflow\_state (field name)

**Step 4:** Populating Workflow State childdoctype

For all values in role permission tool child table

IF idx(position of element in child doctype) == 1

Set role\_name = designation

FOR workflow\_state in [{"state":"Draft","update\_value":"Invoice in Draft State"},{"state":"Verify and Save","update\_value":"Document save"}, {"state":"Cancelled","update\_value":"Document cancelled Note Keeper"}]

FOR each state appends workflow with

State: name of description

Update\_fields: document\_status

Update\_value: state[update\_value]

Allow\_edit: doc.designation\_name

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

IF workflow\_state = Reject and Transfer

Update\_values = Rejected and Transfer

FOR each state appends workflow with

State=name of description

Doc\_status=0

Updates\_field; document\_status

Update\_value: Rejected and Transfer

Allow\_edit: doc.designation\_name

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

FOR(other state in state child doctype)

New\_index=idx+1

list\_state=[{"state":"Verified & Submitted by Note Creator","update\_value":"Submitted by Note Keeper"}]

Get new state of workflow from list\_state

Get updated\_value from list\_state

Role\_name, single\_user, group\_of\_destination will be designation found against line item

FOR each state appends workflow with

State: name of description

Doc\_status: 0

Update\_field: document\_status

update\_value: update\_value

Allow\_edit: doc.designation\_name

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

ELSE

Set flag = “No”

For all values in role permission tool child table

New\_index=idx+1

IF old idx == new\_inedx

Flag =”Yes”

If(description \_of\_state is Null or empty)

state=Append state with “Approved by”+designation(doc)

Update\_value=Append update\_value with “Submitted by”+designation(doc)

ELSE

State=description\_of\_state

Update\_value=description\_of\_State

Role\_name=designation

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

FOR each state appends workflow with

State: name of description

Doc\_status: 0

Update\_field: document\_status

update\_value: update\_value

Allow\_edit: doc.designation\_name

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

IF(Flag=No)

Update\_value=Submitted\_by+designtaion

IF description\_of\_state==None or empty

state=Append state with “Approved by”+designation

Update\_value=Append update\_value with “Submitted by”+designation

ELSE, //for last line item where value not found

State=description\_of\_state

Update\_value=description\_of\_State

Role\_name=designation

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

FOR each state appends workflow with

State: name of description

Doc\_status: 0

Update\_field: document\_status

update\_value: update\_value

Allow\_edit: doc.designation\_name

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

IF state = Pass for payment

Update\_value = Pass for Payment

FOR each state appends workflow with

State: name of description

Doc\_status: 0

Update\_field: document\_status

update\_value: update\_value

Allow\_edit: doc.designation\_name

Single\_user: value from doc

Grouping\_of\_destination: doc.group\_value (A,B,C, etc)

**Step 5:** Populating transition of Child table

For all values in role permission tool child table

If(idx==1):

List\_state = [{"state":"Draft", "action":"Verify and Save", "next\_state":"Verify and Save"}, {"state":"Draft", "action":"Cancel", "next\_state":"Cancelled"}, {"state":"Verify and Save", "action":"Submit", "next\_state":"Verified & Submitted by Note Creator"}]

FOR States in List\_item:

Get state, action, next\_state from List\_state dictionary

FOR each Transition line item append with

State: state from dictionary

Action: action from dictionary

Next\_state: next\_state from dictionary

Allowed: set for next transition(role)

Allow\_self\_approval: 1

New\_index=idx+1

Get all values from role permission child doctype

If old\_idx == new\_idx

List\_state = [{"state":"Verified & Submitted by Note Creator"}]

Name of transition will be state from list\_state

IF description\_of\_state == None or empty

Next\_state=Approved by+designation

ELSE

Next\_state=description of state

Allowed\_by=designation

IF approve==1:

State=name\_of\_transition

Action=Approve

Next\_state=next\_state from dictionary

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

IF reject==1

State=name\_of\_transition

Action=Reject

Next\_state=Cancelled

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

IF reject\_and\_transfer==1

State=name\_of\_transition

Action=Reject and Transfer

Next\_state=next\_state from dictionary

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

ELSE

Flag=”No”

New\_idx=idx+1

Get all values from role permission child doctype

IF old\_idx == new\_idx

Flag=”Yes”

IF description\_of\_state == None or empty

Next\_state=Approved by+designation

ELSE

Next\_state=description of state

Allowed\_by=designation

IF approve==1:

State=name\_of\_transition

Action=Approve

Next\_state=next\_state from dictionary

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

IF reject==1

State=name\_of\_transition

Action=Reject

Next\_state=Cancelled

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

IF reject\_and\_transfer==1

State=name\_of\_transition

Action=Reject and Transfer

Next\_state=next\_state from dictionary

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

IF Flag = No

IF description\_of\_state == None or empty

Next\_state=Approved by+designation

ELSE

Next\_state=description of state

Allowed\_by=designation

IF approve==1:

State=name\_of\_transition

Action=Approve

Next\_state=next\_state from dictionary

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

IF reject==1

State=name\_of\_transition

Action=Reject

Next\_state=Cancelled

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

IF reject\_and\_transfer==1

State=name\_of\_transition

Action=Reject and Transfer

Next\_state=next\_state from dictionary

Append workflow transition table by:

State: state

Action: action

Next\_state: next\_state

Allowed: fetch from user input

Allow self approval: 1

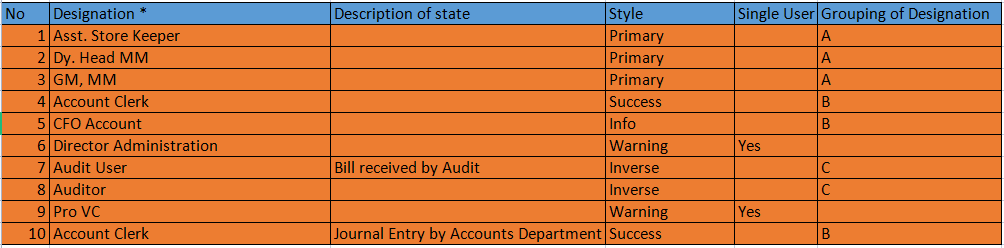
**Step 6:** Save the workflow

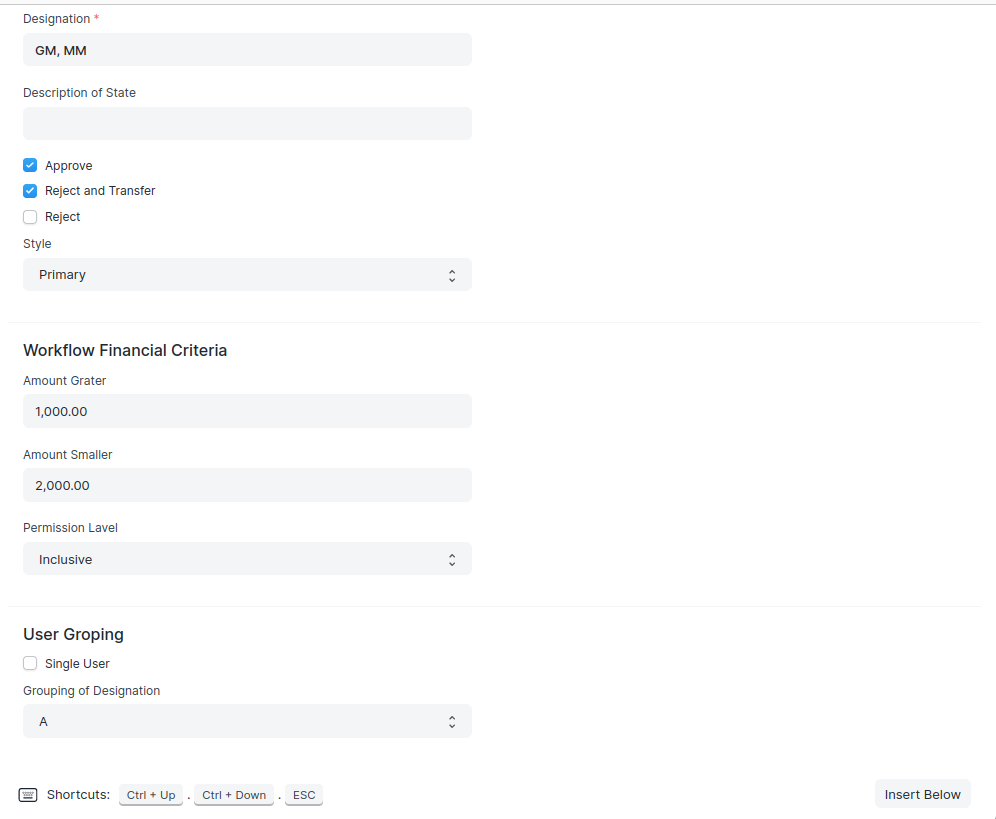
1. **Dynamic allocation of signatures in print format**

**Input:**

* Designation (Link)
* Description of State (Data)
* Single User (Checkbox)
* Group By destination (Select)
* Style (Link)

**Programming Logic:**

****

****

* Get all active Workflows
* Get Workflow name from Workflow Tool Doctype.
* Get Total number of unique Group by destination from Workflow tool child doctype using SQL Query.
* FOR total number of groups:

Get section wise name as Input by the user

FOR Each section append Designation’s signature & remarks having similar Group of designation.

1. **Adding Checkbox for Each Attachment Button**

For Each line items Attachment fields there would be a provision to first click a checkbox after which the attach button or uploading the necessary documents will be appearing.

**Input:**

|  |  |  |
| --- | --- | --- |
| Attachment | |  | | --- | |  | |

**Programming Logic:**

1. Add a checkbox field before every attach button.
2. In the attach button, Go to Edit and under Display depends on write:

eval:name of checkbox==1

**Output:**

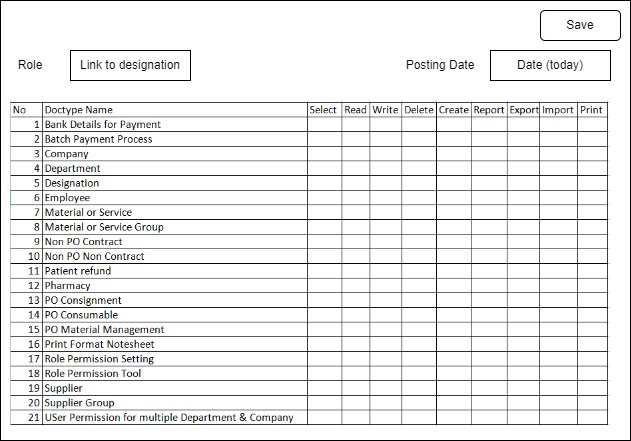
|  |  |  |  |
| --- | --- | --- | --- |
| Attachment | |  | | --- | |  | | Attach |

1. **Role Permission Tool**

**Input:**

* Role
* Posting Date
* Select, Read, Write, Delete, Create, Report, Export, Import permission for respective doctypes.

**Wireframe:**



**Programming Logic:**

* Select the role for which you want to set role permission from a link field.
* The child doctype containing list of all doctype appears in the bottom where permission for Select, Read, Write, Delete, Create, Report, Export, Import, Print will be ticked and saved.
* After save there will be creation of a new instance in Custom Docperm doctype for this role which we have created, if it doesnot exist.
* Else update Custom Docperm for existing values if it exist.

**Output:**

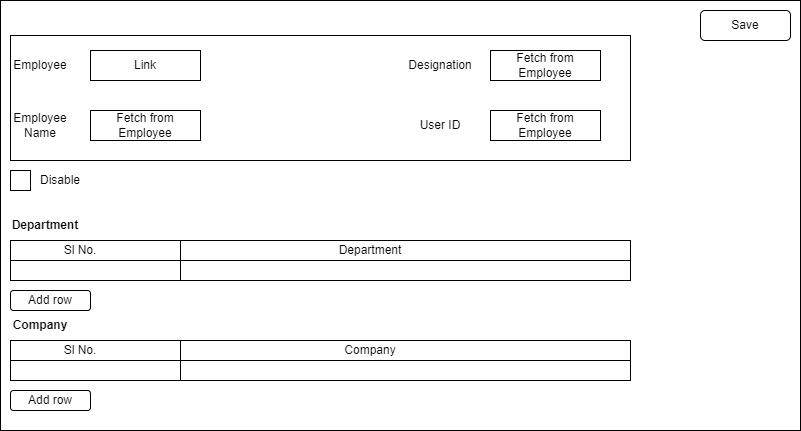
Records in custom docperm for role that we have selected and particular doctype.

1. **User Permission Tool**

**Input:**

* Employee (Link)
* Employee Name (Text)
* Designation (Text)
* User ID (Text)
* Disable(Checkbox)
* Department (Childdoctype)
* Company (Childdoctype)

**Wireframe:**



**Programming Logic:**

* Select the Employee for which you want to create role permission from a link field.
* Select Values for Department and Company from Child doctypes to select multiple values.
* After Save create instance for User permission of the selected employees for current Department and Companies.

**Output:**

* User permission record for the Employee.
* Employee will be able to see data associated with their own Department/ Company.

1. **Print requirement uploading PDF For all Invoices:**

**Requirement:**

While uploading Attachments user should only be able to upload files in pdf format.

Option to upload attachment in Link, Photo and other format must be disabled.

**Programming Logic:**

* File uploader.vue must be brought to our own app and be overridden with code where button for other upload option would be removed and run.
* For uploading only pdf files we can check the files extension while user is uploading it and throw an error incase if it is not a pdf file.

1. **Sending Workspace for Approval/Verification to a third party outside of the workspace**

**Input:**

* Link field for Selecting Employee whom to be transferred
* Remarks
* New Doctype for storing remarks from Outside Employee

**Programming Logic:**

* Create a Link field in Signature Childdoctype where Employee should be selected to whom Notesheet would be transferred.
* Create a new doctype where This transferred Notesheet record would be created to view/Verify and update remarks from the employee who is viewing it outside the workflow.
* The roles and permission must be created for the instance of the remarks that was created to all users in the workflows signature child doctype, so that anyone present in the workflow can view the remarks sent by the outside party.
* As this approval process is going on the Notesheet(Sent for approval) must remain in pending state so that so further changes could be made to it
* After Approval the Employee(outside of workflow)attaches Comment and sent it back to the Authority who have sent it for Approval and the workflow resumes from that state.

1. **Invoice Entry Screen:**

**Requirement:**

Invoice Entry Screen will be provided to the Note Sheet creator for Supplier wise Invoice entry which will be fetched while creating Note-sheet.

**Programming Logic:**

* User Enters Invoice number, Supplier code, Supplier name, Invoice date, Invoice amount, Posting date.
* After Submitting when the Note-sheet creator creates a Note-sheet and Selects a Supplier from the list of supplier that appears, the invoices in which the supplier is present will be filtered and displayed in the child table of that doctype from where the Note-sheet creator select the invoice that they require.
* After the selecting of Invoice in Note-sheet the Note-sheet number, status and type of Note-sheet will be populated in the Invoice Entry Screen.
* Also the Batch payment number and Payment status of Invoice will be updated once the Note-sheet containing the invoice reaches that phase.