**SAP - APPROVAL WORKFLOW**

### 

### 

### 

### 

### 

### **PREPARED FOR**

Rane Madras limited

## **ADMIN MANUAL STRUCTURE**

### **1. Introduction**

# **Purpose of the Application**

# **The SAP Approval Workflow application is designed to streamline and digitize the material request approval process within an organization. It allows users to raise requests and enables multi-level approvals based on user roles and hierarchies.**

# **Target audience of the manual (Admins, IT support, etc.)**

# **Technologies Used**

# **Frontend:** [**React.js**](http://react.js)

# **Backend: Node.js with Express**

Database: MS Sql Server

# 

### **3. Folder Structure and Paths**

#### **Frontend:**

# **Frontend: React (SAP-Frontend)**

* **Computer IP :** **10.51.10.115**

# **Path: D:\Vishnu Priya\SAP\SAP-Frontend**

# **Port: 30001**

# **Start Command: npm start**

# **Build Command: npm run build**

#### **Backend:**

# **Backend: Node.js/Express (SAP-Backend)**

# **Path: D:\Vishnu Priya\SAP\SAP-Backend**

# **Port: 3000**

# **Start Command: npm start**

### **4. Installation & Setup**

#### **Backend Setup:**

# **Navigate to backend folder:cd D:\Vishnu Priya\SAP\SAP-Backend**

# **Install dependencies: npm install**

# **Start the backend: npm start**

# **Frontend Setup:**

# **Navigate to frontend folder:cd D:\Vishnu Priya\SAP\SAP-Frontend**

# **Install dependencies:npm install**

# **Start the frontend: npm start**

# **Build for production: npm run build**

### 

### 

### 

### 

### **5. Application Overview**

#### **Approval Workflow Levels: User Roles & Approval Levels**

The SAP Approval Workflow supports multiple roles across 4 main approval levels. Here's how they're defined:

#### **Level 1 – Requester**

* **Role\_ID:** 1
* **Access:** Can create material requests.

#### **Level 2 – Department/Plant Level Approvers**

* **Role\_IDs:** 2 (PLANT MMD HEAD), 3 (PLANT FINANCE HEAD), 4 (PLANT MRPC)
* **Access:** Can approve/reject L1 requests.

#### **Level 3 – Functional/Plant Heads**

* **Role\_IDs:** 5 (PLANT HEAD), 6 (CORP FINANCE HEAD), 7 (CORP MRPC)
* **Access:** Approves after L2.

#### **Level 4 – Business Head**

* **Role\_ID:** 8
* **Access:** Final approval authority.

#### **Admin / Others**

* **CORP ADMIN (9):** Manages users and roles.
* **Other Roles (10, 11, 14, 15):** Informational or task-based roles, not in core approval loop.

# **Project Creation and Project details API:**

1. Home Page

# **Route path : http://localhost:3001/home/Home**

* + - **Based On Employee**

import { decryptSessionData } from "../../controller/StorageUtils";

1. Dashboard

# **Route path : http://localhost:3001/home/dashboard**

* + - **Based On Employee**

import { getLogin } from "../../controller/Masterapiservice";

import {encryptSessionData,decryptSessionData,}

from "../../controller/StorageUtils";

import { AuthContext } from "../../Authentication/AuthContext";

* **Masters**

## **Module: Company Master**

1. **Frontend Details**

**Description: Add, update and view company master data**

# **Route path:** [**http://localhost:3001/home/company**](http://localhost:3001/home/company)

# **Component path: src/Master/Company.jsx**

**API Service File:**

* **src/controller/CompanyMasterapiservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET /CompanyMaster/get\_details\_Company**
    - **getAdd(data) → POST /CompanyMaster/Get\_Add**
    - **getUpdates(data) → PUT /CompanyMaster/get\_Updates**
    - **User\_Img(data) → POST /CompanyMaster/Upload\_Image**

1. **Backend Details**

**Backend File: server/routes/CompanyMaster.Router.js**

* + - **SQL Tables: Mst\_Company**

**Purpose → Stores master data of companiesSQL**

* + - **Procedures Used:**

1. **GetMst\_Company → Fetch all company master records**
2. **InsertCompany → Insert new company record**
3. **ImageUpload → Update existing company record**
4. **UpdateCompany → Upload company logo image**

## **2. Module:Business Division Master**

1. **Frontend Details**

**Description:Add, update, view business division and link to company**

# **Route path: http://localhost:3001/home/BusinessDivision**

# **Component path: src/Master/BusinessDivision.jsx**

**API Service File:**

* **src/controller/BusinessDivisionMasterapiservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET /BusinessDivisionMaster/get\_details\_BusinessDivision**
    - **getAdd(data) → POST /BusinessDivisionMaster/Get\_Add**
    - **getUpdates(data) → PUT /BusinessDivisionMaster/get\_Updates**
    - **getCompany() → GET /BusinessDivisionMaster/Get\_Company**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/BusinessDivisionMaster.Router.js**

* + - **SQL Tables: Mst\_Business\_Division**
    - **SQL Fetch Other Table: Mst\_Company**

**Purpose → Stores business division master data**

* + - **Procedures Used:**

1. **GetMst\_Business\_Division → Fetch all business division records**
2. **InsertBusinessDivision → Insert new business division**
3. **UpdateBusinessDivision → Update business division info (name, address, status)**
4. **GetActiveCompany → Fetch all active companies for dropdown/listing**

## **3. Module:Plant Master**

1. **Frontend Details**

**Description: The Plant Master module allows you to add, update, and manage plant records.**

**Each plant is linked to a company and includes code, name, and status details.**

# **Route path: http://localhost:3001/home/Department**

# **Component path: src/Master/Plant.jsx**

**API Service File:**

* **src/controller/PlantMasterapiservices.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET /PlantMaster/get\_details\_Plant**
    - **getAdd(data) → POST /PlantMaster/Get\_Add**
    - **getUpdates(data) → PUT /PlantMaster/get\_Updates**
    - **getCompany() → GET /PlantMaster/Get\_Company**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/BusinessDivisionMaster.Router.js**

* + - **SQL Tables: Mst\_Plant**
    - **SQL Fetch Other Table: Mst\_Company**

**Purpose → Stores business division master data**

* + - **Procedures Used:**

1. **GetMst\_Plant → Fetches all plant master records from the Mst\_Plant table**
2. **InsertPlant → Inserts a new plant record including company link, name, and status**
3. **UpdatePlant → Updates existing plant details (name, short name, status); restricts update if plant is inactive**
4. **GetActiveCompany→ Fetches all active companies for use in dropdowns or company association**

## **4. Module:Department Master**

1. **Frontend Details**

**Description: The Department Master module is used to add, update, and manage department records. Each department includes a code, name, and active status. It helps define departments for request and approval workflows.**

# **Route path: http://localhost:3001/home/Department**

# **Component path: src/Master/Department.jsx**

**API Service File:**

* **src/controller/DepartmentMasterapiservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET /DepartmentMaster/get\_details\_Department**
    - **getAdd(data) → POST /DepartmentMaster/Get\_Add**
    - **getUpdates(data) → PUT /DepartmentMaster/get\_Updates**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/DepartmentMaster.Router.js**

* + - **SQL Tables: Mst\_Department**
    - **SQL Fetch Other Table: Mst\_Company**

**Purpose → Stores all department master data**

* + - **Procedures Used:**

1. **GetMst\_Department → Fetches all department records**
2. **InsertDepartment → Inserts a new department (code, name, status, user ID)**
3. **UpdateDepartment → Updates an existing department; prevents update if inactive**

## **5. Module:User Master**

1. **Frontend Details**

**Description: Manage users by adding, updating, and fetching user records including their plant, department, role, user level, and status.**

# **Route path: http://localhost:3001/home/UserMaster**

# **Component path: src/Master/UserMaster.jsx**

**API Service File:**

* **src/controller/UserMasterapiservice.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /UserMaster/get\_details**
    - **getPlants() → GET /UserMaster/Get\_Plants**
    - **getDepartment() → GET /UserMaster/Get\_Department**
    - **getRole() → GET /UserMaster/Get\_Role**
    - **getUserLevel() → GET /UserMaster/Get\_UserLevel**
    - **getAdd(data) → POST /UserMaster/Get\_Add**
    - **getUpdates(data)→ PUT /UserMaster/get\_Updates**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/UserMaster.Router.js**

* + - **SQL Tables: Mst\_User**
    - **SQL Fetch Other Table:** 
      * **Mst\_Company**
      * **Mst\_Plant**
      * **Mst\_Role**
      * **Mst\_UserLevel**
      * **Mst\_Department**

**Purpose → Stores all user master data**

* + - **Procedures Used:**

1. **GetUserDetails → Fetches all user records**
2. **GetUserActivePlants → Fetches all active plants**
3. **GetActiveDepartment → Fetches all active department**
4. **GetActiveRole → Fetches all active roles**
5. **GetActiveUserLevel → Fetches all active user levels**
6. **Insert\_Mst\_User → Inserts a new user record**
7. **UpdateUserMaster → Updates an existing user; blocks update if inactive**

## **6. Module:Role Master**

1. **Frontend Details**

**Description: Manages roles by allowing add, update, and fetch operations with role name and active status to control user permissions.**

# **Route path: http://localhost:3001/home/Role**

# **Component path: src/Master/Role.jsx**

**API Service File:**

* **src/controller/Roleapiservices.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /RoleMaster/get\_details\_Role**
    - **getAdd() → POST /RoleMaster/Get\_Add**
    - **getUpdates()→ PUT /RoleMaster/get\_Updates**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/RoleMaster.Router.js**

* + - **SQL Tables: Mst\_Role**

**Purpose → Stores role master data including role names and their active status to manage user permissions.**

* + - **Procedures Used:**

1. **GetMst\_Role → Fetches all role records**
2. **Insert\_Mst\_Role → Inserts a new role**
3. **Insert\_Mst\_Role → Updates an existing role, prevents updates if inactive**

## **7. Module:Material Master**

1. **Frontend Details**

**Description: Manages materials by allowing add, update, and fetch operations with material details such as material name, type, plant, and status.**

# **Route path: http://localhost:3001/home/Material**

# **Component path: src/Master/Material.jsx**

**API Service File:**

* **src/controller/Masterapiservice.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /Master/get\_details**
    - **getAdd() → POST /Master/Get\_Add**
    - **getUpdates()→ PUT /Master/get\_Updates**
    - **getPlants() → GET /Master/Get\_Plants**
    - **getMaterialType() → GET /Master/Get\_Material\_Type**
    - **MaterialMaster(data)→ POST /Master/File**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/MaterialMaster.Router.js**

* + - **SQL Tables: Mst\_Material**
    - **SQL Fetch Other Table:** 
      * **Mst\_Material\_Type**
      * **Mst\_Plant**

**Purpose → Stores material information including codes, types, and plant associations, managing inventory and procurement processes.**

* + - **Procedures Used:**

1. **GetActiveMaterialInfo → Fetches all active material records**
2. **InsertMstMaterial → Inserts a new material record**
3. **UpdateMaterial → Updates existing material details**
4. **GetSupvCode → Retrieves supervisor codes based on PlantCode**
5. **GetActiveMaterialType → Fetches all active material types**
6. **GetActivePlants → Fetches all active plants**
7. **UploadMaterial1 → Handles material-related file uploads**

## **8. Module:Vendor Master**

1. **Frontend Details**

**Description: The Vendor Master module allows adding, updating, and fetching vendor information including vendor code, name, address, associated plant, and active status..**

# **Route path: http://localhost:3001/home/Vendor**

# **Component path: src/Master/Vendor.jsx**

**API Service File:**

* **src/controller/VendorMasterapiservice.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /VendorMaster/get\_details\_Vendor**
    - **getAdd(data)→ POST /VendorMaster/Get\_Add**
    - **getUpdates(data) → PUT /VendorMaster/get\_Updates**
    - **getPlants()→ GET /VendorMaster/Get\_Plants**
    - **VendorMaster(data)→ POST /VendorMaster/File**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/VendorMaster.Router.js**

* + - **SQL Tables: Mst\_Vendor**
    - **SQL Fetch Other Table:** 
      * **Mst\_Plant**

**Purpose → Stores vendor details including vendor code, name, address, plant association, and status, enabling supplier management across plants.**

* + - **Procedures Used:**

1. **GetMst\_Vendor → Fetches all vendor records**
2. **InsertVendor → Inserts a new vendor with plant, code, name, and address**
3. **UpdateVendor → Updates existing vendor information**
4. **GetActivePlants → Fetches all active plants**
5. **UploadVendor → Handles file upload and batch insert for vendor data**

## **9. Module:Customer Master**

1. **Frontend Details**

**Description: The Customer Master module manages customer data by enabling create, update, view, and bulk upload operations. Each record stores customer code, name, address, plant code, and active status.**

# **Route path: http://localhost:3001/home/Customer**

# **Component path: src/Master/Customer.jsx**

**API Service File:**

* **src/controller/CustomerMasterapiservice.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /CustomerMaster/get\_details\_Customer**
    - **getAdd(data)→ POST /CustomerMaster/Get\_Add**
    - **getUpdates(data) → PUT /CustomerMaster/get\_Updates**
    - **getPlants()→ GET /CustomerMaster/Get\_Plants**
    - **CustomerMaster(data)→ POST /CustomerMaster/File**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/CustomerMasterMaster.Router.js**

* + - **SQL Tables: Mst\_Customer**
    - **SQL Fetch Other Table:** 
      * **Mst\_Plant**

**Purpose → Stores customer master data including code, name, address, and associated plant. This supports customer-linked workflows in supply chain, sales, and delivery operations.**

* + - **Procedures Used:**

1. **GetMst\_Customer → Fetches all customer records**
2. **InsertCustomer → Inserts a new customer record**
3. **UpdateCustomer → Updates existing customer information**
4. **GetActivePlants → Fetches all active plants**
5. **UploadCustomer → Handles file upload and inserts customer data in bulk**

## **10. Module:Storage Location Master**

1. **Frontend Details**

**Description: The Storage Location Master module manages the creation, update, and mapping of storage locations to supervisors. Each storage location is linked to a plant, has a unique 4-digit code, and can be associated with multiple supervisors.**

# **Route path: http://localhost:3001/home/StorageLocation**

# **Component path: src/Master/StorageLocation.jsx**

**API Service File:**

* **src/controller/StorageLocationapiservice.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /StorageLocation/get\_details**
    - **getAdd(data)→ POST /StorageLocation/Get\_Add**
    - **getUpdates(data) → PUT /StorageLocation/get\_Updates**
    - **getPlants()→ GET /StorageLocation/Get\_Plants**
    - **getSupvCode(plantId)→ GET /StorageLocation/Get\_SupvCode?PlantCode={id}**
    - **getSupvMappingsBySLocId(id) → GET /StorageLocation/supv-mapping/{SLoc\_ID}**
    - **MappingData()→ GET /StorageLocation/Get\_SupvCode\_Mappings**
    - **getStorageDownload()→ GET /StorageLocation/get\_StorageDownload**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/StorageLocationMaster.Router.js**

* + - **SQL Tables: Mst\_Storage\_Location**
    - **SQL Fetch Other Table:** 
      * **Mst\_Plant**
      * **Mst\_SupvCode\_Mapping**

**Purpose → Maintains records of storage locations and their supervisor mappings. Each location is plant-specific and supports mapping to one or more supervisors for operational workflows like inventory management.**

* + - **Procedures Used:**

1. **Insert\_Mst\_StorageLocation → Inserts a new storage location and returns the Sloc\_ID**
2. **UpdateStorageLocation → Updates the storage location details**
3. **GetSupvCode → Fetches available supervisor codes by PlantCode**
4. **Get\_SupvCode\_Mappings → Retrieves all supervisor-location mappings**
5. **GetStorageDownloadData → Used for downloading complete storage location dataset**
6. **Inline Query (by SLoc\_ID)→ Gets all active supervisor mappings for a specific storage location**
7. **Custom Query (Deactivate)→ Marks existing mappings as inactive before remapping**
8. **Custom Query (Insert/Update Mapping)→ Inserts new or reactivates existing supervisor mappings for a location**

## **11. Module:Movement Type Master**

1. **Frontend Details**

**Description: Manages movement types by allowing operations to add, update, and fetch movement codes and names used in inventory or material tracking.**

# **Route path: http://localhost:3001/home/Movement\_Type**

# **Component path: src/Master/MovementType.jsx**

**API Service File:**

* **src/controller/MovementType.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /MovementType/get\_details**
    - **getAdd(data)→ POST /MovementType/Get\_Add**
    - **getUpdates(data) → PUT /MovementType/get\_Updates**
    - **getPlants()→ GET /MovementType/Get\_Plants**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/MovementType.RouterMaster.js**

* + - **SQL Tables: Mst\_Movement\_Type**

**Purpose → Stores movement type definitions, including code, name, and active status, used for inventory tracking, stock movement, and reporting.**

* + - **Procedures Used:**

1. **GetMovementType → Fetches all movement type records**
2. **Insert\_Mst\_MovementType → Inserts a new movement type record**
3. **UpdateMovementType → Updates an existing movement type**

## **12. Module:Movement List Item Master**

1. **Frontend Details**

**Description: Manages Movement List Items, allowing operations to add, update, and view records. Movement list items are subcategories of a movement type, used for detailed tracking.**

# **Route path: http://localhost:3001/home/MVT\_LIST\_ITEM**

# **Component path: src/Master/MovementListItem.jsx**

**API Service File:**

* **src/controller/MovementListItem.jsx**
  + **API Functions Used** 
    - **getdetails() → GET /MovementListItem/get\_details**
    - **getAdd(data)→ POST /MovementListItem/Get\_Add**
    - **getUpdates(data) → PUT /MovementListItem/get\_Updates**
    - **getActiveMovementType()→ GET /MovementListItem/Get\_MovementType**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File:**

**server/routes/MovementListItemMaster.Router.js**

* + - **SQL Tables: Mst\_Movement\_List\_Item**
    - **SQL Fetch Other Table:** 
      * **Mst\_Movement\_Type**

**Purpose → Stores subcategories of movement types, such as movement reasons or sub-movements, for more granular tracking and classification.**

* + - **Procedures Used:**

1. **GetMovementListItem→ Fetch all movement list item records**
2. **Insert\_Mst\_MovementListItem→ Insert a new movement list item record**
3. **UpdateMovementListItem→ Update existing movement list item**
4. **GetActiveMovementType → Fetch all active movement types for selection dropdowns**

## **13. Module:Cost Center Master**

1. **Frontend Details**

**Description: Manage Cost Centers linked to Plants. Supports listing, adding, and updating cost centers.**

# **Route path: http://localhost:3001/home/CostCenter**

# **Component path: src/Master/CostCenter.jsx**

**API Service File:**

* **src/controller/CostCenterapiservices.js**
  + **API Functions Used** 
    - **getdetails() → GET /CostCenter/get\_details**
    - **getPlants()→ GET /CostCenter/Get\_Plants**
    - **getAdd(data) → POST /CostCenter/Get\_Add**
    - **getUpdates(data)→ PUT /CostCenter/get\_Updates**

**2. Backend Details**

**Backend File: server/routes/CostCenterMaster.Router.js**

* + - **SQL Tables: Mst\_Cost\_Center**
    - **SQL Fetch Other Table:** 
      * **Mst\_Plant**

**Purpose → The Cost Center Master module manages plant-specific cost centers used for tracking and controlling departmental expenses. It supports budgeting, financial reporting, and accurate cost allocation across operations.**

* + - **Procedures Used:**

1. **GetCostCenter → Fetch all cost center records**
2. **GetUserActivePlants → Fetch all active plants**
3. **Insert\_Mst\_Cost\_Center → Insert new cost center**
4. **UpdateCostCenter → Update existing cost center**

## **14. Module: SupvCode Master**

1. **Frontend Details**

**Description: Manage Supervisor Codes (SupvCode) linked to specific plants. Supports listing, adding, and updating supervisor records with lead time settings.**

# **Route path: http://localhost:3001/home/SupvCode**

# **Component path: src/Master/SupvCode.jsx**

**API Service File:**

* **src/controller/SupvCodeMasterapiservice.js**
  + **API Functions Used** 
    - **getdetails() → GET /SupvCodeMaster/get\_details**
    - **getAdd(data) → POST /SupvCodeMaster/Get\_Add**
    - **getPlants() → GET /SupvCodeMaster/Get\_Plants**
    - **getUpdates(data)→ PUT /SupvCodeMaster/get\_Updates**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/SupvCodeMaster.js**

* + - **SQL Tables: Mst\_SupvCode**
    - **SQL Fetch Other Table:** 
      * **Mst\_Plant**

**Purpose → The SupvCode Master maintains supervisor code details for each plant, including lead time settings. It ensures correct mapping of supervisors to support plant-specific operational planning and oversight.**

* + - **Procedures Used:**

1. **GetSupvCodeMaster → Fetch all supervisor code records**
2. **GetUserActivePlants → Fetch all active plant records**
3. **Insert\_Mst\_SupvCode → Add a new supervisor code**
4. **UpdateMstSupvCode → Update existing supervisor code**

## **15. Module: Module Master**

1. **Frontend Details**

**Description: Manage module names associated with departments and plants. Allows listing, adding, and updating modules.**

# **Route path: http://localhost:3001/home/Module**

# **Component path: src/Master/ModuleMaster.jsx**

**API Service File:**

* **src/controller/ModuleMasterapiservice.js**
  + **API Functions Used** 
    - **getdetails() → GET /ModuleMaster/get\_details**
    - **getPlants() → GET /ModuleMaster/Get\_Plants**
    - **getDepartment() → GET /ModuleMaster/Get\_Department**
    - **getAdd(data) → POST /ModuleMaster/Get\_Add**
    - **getUpdates(data)→ PUT /ModuleMaster/get\_Updates**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/ModuleMaster.js**

* + - **SQL Tables: Mst\_Module**
    - **SQL Fetch Other Table:** 
      * **Mst\_Plant**
      * **Mst\_Department**

**Purpose → The Module Master manages modules linked to departments and plants for streamlined operations and role-based access control.**

* + - **Procedures Used:**

1. **GetModule → Fetch all module records**
2. **GetUserActivePlants → Get active plants list**
3. **GetActiveDepartment → Get active departments**
4. **Insert\_Mst\_Module → Insert a new module**
5. **UpdateMst\_Module → Update existing module**

## **15. Module:Admin Role Master →User Screen Permission**

1. **Frontend Details**

**Description: Add, view Role Permission data**

* **Route path:****http://localhost:3001/home/Role/Roleid**
* **Component path: src/Master/Admin.jsx**

**API Service File:**

* **src/controller/AdminMasterapiservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET /AdminMaster/menu\_permission?Role\_id=${role}**
    - **getAdd(data) → POST /AdminMaster/AddAccessMenu**
    - **getScreenType(data) → GET**

**/AdminMaster/ScreenType?Role\_id=${roleIdNo}**

* + - **getScreenName(data) → GET /AdminMaster/ScreenName?Role\_id=${roleIdNo}&Screen\_Type=${Screen\_Type}**

1. **Backend Details**

**Backend File: server/routes/AdminMaster.Router.js**

* **SQL Tables: Mst\_Access**
* **SQL Fetch Other Table: Mst\_Screen**

**Purpose → Stores master data of Role Screen Permission**

* **Procedures Used:**

1. **GetAccessDetails → Fetch all Screen Permission records**
2. **AddAccess→ Insert new Screen Permission**
3. **GetScreenType→Fetch ScreenType Records**
4. **GetSubMenus→Fetch ScreenName of the ScreenType**

## **17. Module: Submenu Page (Inside Role Page)**

1. **Frontend Details**

**Description:Handles screen-level access (submenu permissions) under a selected role. Allows assigning and removing submenu access for each role, improving granular control over role-based UI rendering and security.**

# **Route path: http://localhost:3001/home/Role/:roleId?role=Admin&menu\_name=ScreenType**

# **Component path: src/Master/Submenu.jsx**

**API Service File:**

* **src/controller/AdminMasterapiservice.jsx**
  + **API Functions Used** 
    - **getdetailssub() → GET /AdminMaster/SubMenuNames?Role\_id={role}&menuName={menu}**
    - **get\_Sub\_Menu\_List() → GET /AdminMaster/SubMenuList?Role\_id={role}&menuName={menu}**
    - **AddMenuAccess() → POST /AdminMaster/AddAccessMenu**
    - **Delete\_Menu()→ PUL /AdminMaster/Delete\_Access?Access\_Id={id}&EmployeeId={uid}**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/AdminMaster.js**

* + - **SQL Tables: Mst\_Screen**
    - **SQL Fetch Other Table:** 
      * **Mst\_Access**
      * **Mst\_Role**
    - **Procedures Used:**

1. **GetModule → Retrieves all menu permissions by role**
2. **SubMenusRole → Returns available submenus not yet assigned to the role**
3. **GetAvailableMenu → Fetches current submenu names under a menu for the role**
4. **AddAccess → Adds submenu/screen access to the role**
5. **DeleteAccess → Soft deletes (revokes) submenu access**

**18. Module:Line Master**

1. **Frontend Details**

**Description: Add, view,Edit**

**Route path:****http://localhost:3001/home/Line**

* **Component path: src/Master/Line.jsx**

**API Service File:**

* **src/controller/AdminMasterapiservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET**

**/LineMaster/get\_details**

* + - **getAdd(data) → POST**

**/LineMaster/Get\_Add**

* + - **getUpdates(data) → PUT**

**/LineMaster/get\_Updates**

* + - **getDepartment(data) → GET /LineMaster/Get\_Department**
    - **getSupvCode(data) → GET /LineMaster/Get\_SupvCode?Plant\_ID=${Plant\_ID}**

* + - **getModule(data) → GET /LineMaster/Get\_Module**

* + - **getPlants(data) → GET /LineMaster/Get\_Plants**

1. **Backend Details**

**Backend File: server/routes/AdminMaster.Router.js**

* **SQL Tables: Mst\_Line**
* **SQL Fetch Other Table: Mst\_Module,Mst\_SupvCode,Mst\_Department,Mst\_Plant**

**Purpose → Stores master data of Line**

* **Procedures Used:**

**1.GetLine → Fetch all Line Records**

**2.GetUserActivePlants→Fetch Active Plants Records**

**3.GetActiveDepartment→Fetch Active Department Records**

**4.Insert\_Mst\_Line→Insert Line Records**

**5.UpdateMst\_Line→Updated Existing Record in Line**

**6.GetSupvCode→Fetch SupvCode Records**

**7.GetModuleForLine→Fetch Module Records**

**—--------------------------------------------------**

* **Report**

## **Module: Report 309 – Excel Export**

1. **Frontend Details**

**Description: Generates and downloads Excel report for 309 movement transactions within a selected date range. Includes date validations and styled Excel output.**

# **Route path: http://localhost:3001/home/Report3**

# **Component path: src/Reports/Report3.jsx**

**API Service File:**

* **src/controller/Report309apiservice.jsx**
  + **API Functions Used** 
    - **getTransactionData(from, to) → GET /Report3/download\_data**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Report309.Router.js**

* + - **SQL Tables: Trn\_SapTransfer\_Records**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetExistingTrnSap309Movt), and returns 309 transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_309\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetExistingTrnSap309Movt → Retrieves movement transactions in the 309 category based on date range.**

## **Module: Report 201– Excel Export**

1. **Frontend Details**

**Description: Generates and downloads Excel report for 201 movement transactions within a selected date range. Includes date validations and styled Excel output.**

# **Route path: http://localhost:3001/home/Report4**

# **Component path: src/Reports/Report4.jsx**

**API Service File:**

* **src/controller/Report201apiservice.jsx**
  + **API Functions Used** 
    - **getTransactionData(from, to) → GET /Report4/download\_data**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Report201.Router.js**

* + - **SQL Tables: Trn\_SapTransfer\_Records**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetExistingTrnSap201Movt), and returns 201 transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_201\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetExistingTrnSap201Movt → Retrieves movement transactions in the 201 category based on date range.**

## **Module: Report 202– Excel Export**

1. **Frontend Details**

**Description: Generates and downloads Excel report for 202 movement transactions within a selected date range. Includes date validations and styled Excel output.**

# **Route path: http://localhost:3001/home/Report5**

# **Component path: src/Reports/Report5.js**

**API Service File:**

* **src/controller/Report202apiservice.jsx**
  + **API Functions Used** 
    - **getTransactionData(from, to) → GET /Report5/download\_data**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Report202.Router.js**

* + - **SQL Tables: Trn\_SapTransfer\_Records**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetExistingTrnSap202Movt), and returns 202 transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_202\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetExistingTrnSap202Movt → Retrieves movement transactions in the 202 category based on date range.**

## **Module: Report 551– Excel Export**

1. **Frontend Details**

**Description: Generates and downloads Excel report for 551 movement transactions within a selected date range. Includes date validations and styled Excel output.**

# **Route path: http://localhost:3001/home/Report6**

# **Component path: src/Reports/Report6.jsx**

**API Service File:**

* **src/controller/Report551apiservice.jsx**
  + **API Functions Used** 
    - **getTransactionData(from, to) → GET /Report6/download\_data**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Report551.Router.js**

* + - **SQL Tables: Trn\_SapTransfer\_Records**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetExistingTrnSap551Movt), and returns 551 transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_551\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetExistingTrnSap551Movt → Retrieves movement transactions in the 551 category based on date range.**

## **Module: Report 311– Excel Export**

1. **Frontend Details**

**Description: Generates and downloads Excel report for 311 movement transactions within a selected date range. Includes date validations and styled Excel output.**

# **Route path: http://localhost:3001/home/Report7**

# **Component path: src/Reports/Report7.jsx**

**API Service File:**

* **src/controller/Report311apiservice.jsx**
  + **API Functions Used** 
    - **getTransactionData(from, to) → GET /Report7/download\_data**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Report311.Router.js**

* + - **SQL Tables: Trn\_SapTransfer\_Records**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetExistingTrnSap311Movt), and returns 311 transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_311\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetExistingTrnSap311Movt → Retrieves movement transactions in the 311 category based on date range.**

## **Module: Report Rs.1 – Excel Export**

1. **Frontend Details**

**Description: Generates and downloads Excel report for Rs.1 movement transactions within a selected date range. Includes date validations and styled Excel output.**

# **Route path: http://localhost:3001/home/Report8**

# **Component path: src/Reports/Report8.jsx**

**API Service File:**

* **src/controller/ReportConversionapiservice.jsx**
  + **API Functions Used** 
    - **getTransactionData(from, to) → GET /Report8/download\_data**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/ReportRs1.Router.js**

* + - **SQL Tables: Trn\_SapTransfer\_Records**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetExistingConversionRs1Movt), and returns Rs.1 transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_Rs.1\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetExistingConversionRs1 → Retrieves movement transactions in the Rs.1 category based on date range.**

## **Module: Emergency Procurement Report – Excel Export**

1. **Frontend Details**

**Description: Generates and downloads Excel report for Emergency movement transactions within a selected date range. Includes date validations and styled Excel output.**

# **Route path: http://localhost:3001/home/Report2**

# **Component path: src/Reports/Report2.jsx**

**API Service File:**

* **src/Reports/ReportEmergency.jsx**
  + **API Functions Used** 
    - **getEmergencyReport(from, to)→ GET /Inward/downloadReportEmergency**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Inward.Router.js**

* + - **SQL Tables: Trn\_Inward\_Old\_Invoice**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetInwardInvoicesByDateForEmergencyReport ), and returns Emergency transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_Emergency\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetInwardInvoicesByDateForEmergencyReport → Retrieves movement transactions in the Emergency category based on date range.**

## **Module: Inward of Old Invoice Report – Excel Export**

1. **Frontend Details**

**Description: Exports a detailed list of *Inward Invoices* over a selected date range. Includes strict validation (no future dates, no From > To errors). Returns structured data for client-side Excel export.**

# **Route path: http://localhost:3001/home/Report1**

# **Component path: src/Reports/Report1.jsx**

**API Service File:**

* **src/controller/ReportInwardService.jsx**
  + **API Functions Used** 
    - **getInwardReport(from, to)→ GET /Inward/downloadReportInward**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Inward.Router.js**

* + - **SQL Tables: Trn\_Inward\_Old\_Invoice**

**Purpose → Validates date inputs, runs a SQL stored procedure (GetInwardInvoicesByDateForInwardReport ), and returns Emergency transaction data for Excel export.**

* + - **Excel Formatting:**

**Named: Trn\_Emergency\_Movement\_List.xlsx.**

* + - **Procedures Used:**

1. **GetInwardInvoicesByDateForInwardReport → Retrieves movement transactions in the Inward category based on date range.**

**—--------------------------------------------------**

* **Approved Reports**

## **Module:Approval Replorts– Excel Export**

1. **Frontend Details**

**Description: Allows users to download approved report data (e.g., Stock Postings, Scrap, Inward Invoices, Emergency Procurement, etc.) as Excel files based on selected date ranges. Includes dropdowns for report types, date range inputs, and validates all fields before generating styled .xlsx exports.**

# **Route path: http://localhost:3001/home/ApprovalReports**

# **Component path: src/Reports/ApprovedReports.jsx**

**API Service File:**

* **src/controller/Report309apiservice.jsx**
  + **API Functions Used** 
    - **getReportOptions() → GET /ApprovedReports/Get\_Reports**
    - **getReportData(reportType, fromDate, toDate) → GET /ApprovedReports/getReportData**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/ApprovedReports.Router.js**

* + - **SQL Tables:** 
      * **Trn\_SapTransfer\_Records**
      * **Trn\_Inward\_Old\_Invoice**
    - **Excel Formatting:**

**Named: Based On The Movement**

**201\_DATA\_Report.xlsx**

**202\_DATA\_Report.xlsx**

**309\_DATA\_Report.xlsx**

**311\_DATA\_Report.xlsx**

**551\_DATA\_Report.xlsx**

**Conversion\_RS1\_Report.xlsx**

**EMRY\_INV\_DATA\_Report.xlsx**

**OLD\_INV\_DATA\_Report.xlsx**

* + - **Procedures Used:**

**Procedures Used Dynamically (Based on reportType)  
1.GetExistingTrn201Movt**

**2.GetExistingTrn202Movt**

**3.GetExistingTrnSap309Movt  
4.GetExistingTrn311Movt  
5.GetExistingTrn551Movt  
6.GetExistingConversionRS1  
7.GetInwardInvoicesByDateForEmergencyReport  
8.GetInwardInvoicesByDateForInwardReport**

**—--------------------------------------------------**

* **Transaction(dashboard)**

## **1. Module:309 Movement Transaction**

**1. Frontend Details**

**Description: Handles submission, editing, viewing, and downloading of Material-to-Material conversion records (309 movement). Enables export of Excel data, supports approval resubmissions, and displays approval status by Doc\_ID.**

* **Route path: http://localhost:3001/home/Partno**
* **Component path: src/Dashboard/Partno.jsx**

**API Service File:**

* **src/controller/transactionapiservice.js**
  + **API Functions Used** 
    - **getTransactionData() → GET /transaction/download\_data**
    - **DownloadAllExcel() → GET /transaction/download?docId=<id>**
    - **Movement309() → POST /transaction/File**
    - **Movement309ReUpload() → POST /transaction/FileReupload**
    - **getdetails()→ GET /transaction/get\_details?userid=<user>**
    - **Edit309Record()→ POST /transaction/Edit309Record**
    - **get309ApprovalView() → GET /transaction/View309ApprovalStatus**
    - **getresubmit()→ POST /transaction/HandleResubmitAction**
    - **getPlants()→ GET /transaction/Get\_Plants**
    - **getMaterial() → GET /transaction/Get\_Material**
    - **getSLoc()→ GET /transaction/Get\_StorageLocation**
    - **getValuationType()→ GET /transaction/Get\_ValuationType**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/transaction.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Material**
  + **Mst\_Material\_Type**
  + **Mst\_Storage\_Location**
  + **Mst\_Valuation\_Type**
  + **Mst\_Movement\_List\_Item**
  + **Mst\_Movement\_Type**
  + **Mst\_User**

**Purpose → Raised Request and Get Approval for**

**SAP – 309 Movement Transaction.**

* **Procedures Used:**

1. **InsertTrn309Movt→ Handles insertion of new 309 movement transactions**
2. **Edit309Record→ Updates an existing 309 transaction record**
3. **UploadTrn309SapTransferRecord → Uploads file records for 309 movement**
4. **GetExistingTrnSap309Movt → Downloads 309 data within a selected date range (Excel export)**
5. **GetExistingDocIDTrn309Movt → Downloads specific Doc ID's data as Excel**
6. **ResubmitApproval→ Handles resubmission of approval workflow**
7. **View\_PartnoApproval\_Status→ Fetches approval status of 309 movement by Doc ID**
8. **GetView\_309ApprovalStatus\_ByID → Gets detailed view by approval Doc ID**
9. **GetActiveStorageLocation → Lists available storage locations**
10. **GetActiveMaterial → Lists material master data**
11. **GetActiveValuationType → Fetches valuation types for dropdowns**
12. **GetActivePlants → Fetches plant list**
13. **GetTrnSap\_309Movt → Fetches details of existing 309 movements (possibly full list view)**

## **2. Module:201 Movement Transaction – Stock Posting (-)**

**1. Frontend Details**

**Description: This module allows users to raise, track, and download SAP 201 movement transactions (Stock posting for cost center consumption).  
 Users can either:**

* **Upload transactions via an Excel file.**
* **Or submit requests manually through a form interface.**

**Transactions are routed for approval, and the approval status can be viewed using the Doc\_ID.**

* **Route path: http://localhost:3001/home/Stock201**
* **Component path: src/dashboard/Stock201.jsx**

**API Service File:**

* **src/controller/Movement201apiservice.js**
  + **API Functions Used** 
    - **Movement201(data) → POST /Movement201/File**
    - **Movement201Reupload() → POST /Movement201/FileReupload?Doc\_ID=<id>**
    - **getdetails(user) → GET /Movement201/get\_details?userid=<user>**
    - **Edit201Record(data) → POST /Movement201/Edit201Record**
    - **getresubmit(data)→ POST /Movement201/HandleResubmitAction201**
    - **getCancel(data)→ POST /Movement201/HandleCancelAction201?Doc\_ID**
    - **get201ApprovalView(id) → GET /Movement201/View201ApprovalStatus?Doc\_ID=<id>**
    - **getTransactionData()→ GET /Movement201/download\_data?From=...&to=...**
    - **DownloadAllExcel(id)→ GET /Movement201/download\_data?Doc\_ID=<id>**
    - **getPlants() → GET /Movement201/Get\_Plants**
    - **getMaterial()→ GET /Movement201/Get\_Material**
    - **getSLoc()→ GET /Movement201/Get\_StorageLocation**
    - **getMovement()→ GET /Movement201/Get\_MovementType**
    - **getReasonForMovement()→ GET /Movement201/Get\_ReasonForMovementType**
    - **getCostCenter()→ GET /Movement201/Get\_CostCenter**
    - **getValuationType()→ GET /Movement201/Get\_ValuationType**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Movement201.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Material**
  + **Mst\_Material\_Type**
  + **Mst\_Storage\_Location**
  + **Mst\_Valuation\_Type**
  + **Mst\_Movement\_List\_Item**
  + **Mst\_Movement\_Type**
  + **Mst\_User**
  + **Mst\_CostCenter**

**Purpose → Raised Request and Get Approval for**

**SAP – 201 Movement Transaction.**

* **Procedures Used:**

1. **InsertTrn201Movt→ Handles insertion of new 201 movement transactions**
2. **Edit201Record→ Updates a rejected or draft 201 record**
3. **UploadTrn201SapTransferRecord → Handles bulk upload via Excel for 201 movement**
4. **GetExistingTrn201Movt → Retrieves filtered 201 movement records for download(Excel export)**
5. **GetExistingDocIDTrn201Movt → Downloads data of specific Doc\_ID (Excel export)**
6. **ResubmitApproval → Resubmits a rejected approval request**
7. **CancelApproval201 → Cancels an approval request**
8. **View\_Stock201Approval\_Status → Gets approval status for given Doc\_ID**
9. **GetTrnSap201Movt → Fetches detailed 201 movement record list**
10. **GetActiveStorageLocation → Fetches active storage location list**
11. **GetActiveMaterial → Returns all active materials**
12. **GetActiveValuationType → Lists valuation types (dropdown)**
13. **GetActivePlants → Returns available plants**
14. **GetActiveMovementType → Lists movement types (201)**
15. **Get201ReasonForMovement → Lists valid reasons for 201 transaction**
16. **GetActiveCostCentertType → Lists cost center values**

## **3. Module:202 Movement Transaction – Stock Posting (+)**

**1. Frontend Details**

**Description: This module allows users to raise, track, and download SAP 202 movement transactions (Stock posting for cost center consumption).  
 Users can either:**

* **Upload transactions via an Excel file.**
* **Or submit requests manually through a form interface.**

**Transactions are routed for approval, and the approval status can be viewed using the Doc\_ID.**

* **Route path: http://localhost:3001/home/Stock202**
* **Component path: src/dashboard/Stock202.jsx**

**API Service File:**

* **src/controller/Movement202apiservice.js**
  + **API Functions Used** 
    - **Movement202(data) → POST /Movement202/File**
    - **Movement202Reupload() → POST /Movement202/FileReupload?Doc\_ID=<id>**
    - **getdetails(user) → GET /Movement202/get\_details?userid=<user>**
    - **Edit202Record(data) → POST /Movement202/Edit202Record**
    - **getresubmit(data)→ POST /Movement202/HandleResubmitAction202**
    - **getCancel(data)→ POST /Movement202/HandleCancelAction202?Doc\_ID**
    - **get202ApprovalView(id) → GET /Movement202/View202ApprovalStatus?Doc\_ID=<id>**
    - **getTransactionData()→ GET /Movement202/download\_data?From=...&to=...**
    - **DownloadAllExcel(id)→ GET /Movement202/download\_data?Doc\_ID=<id>**
    - **getPlants() → GET /Movement202/Get\_Plants**
    - **getMaterial()→ GET /Movement202/Get\_Material**
    - **getSLoc()→ GET /Movement202/Get\_StorageLocation**
    - **getMovement()→ GET /Movement202/Get\_MovementType**
    - **getReasonForMovement()→ GET /Movement202/Get\_ReasonForMovementType**
    - **getCostCenter()→ GET /Movement202/Get\_CostCenter**
    - **getValuationType()→ GET /Movement202/Get\_ValuationType**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Movement202.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Material**
  + **Mst\_Material\_Type**
  + **Mst\_Storage\_Location**
  + **Mst\_Valuation\_Type**
  + **Mst\_Movement\_List\_Item**
  + **Mst\_Movement\_Type**
  + **Mst\_User**
  + **Mst\_CostCenter**

**Purpose → Raised Request and Get Approval for**

**SAP – 202 Movement Transaction.**

* **Procedures Used:**
* **InsertTrn202Movt→ Handles insertion of new 202 movement transactions**
* **Edit202Record→ Updates a rejected or draft 202 record**
* **UploadTrn202SapTransferRecord → Handles bulk upload via Excel for 202 movement**
* **GetExistingTrn202Movt → Retrieves filtered 202 movement records for download(Excel export)**
* **GetExistingDocIDTrn202Movt → Downloads data of specific Doc\_ID (Excel export)**
* **ResubmitApproval → Resubmits a rejected approval request**
* **CancelApproval202 → Cancels an approval request**
* **View\_Stock202Approval\_Status → Gets approval status for given Doc\_ID**
* **GetTrnSap202Movt → Fetches detailed 202 movement record list**
* **GetActiveStorageLocation → Fetches active storage location list**
* **GetActiveMaterial → Returns all active materials**
* **GetActiveValuationType → Lists valuation types (dropdown)**
* **GetActivePlants → Returns available plants**
* **GetActiveMovementType → Lists movement types (202)**
* **Get202ReasonForMovement → Lists valid reasons for 202 transaction**
* **GetActiveCostCentertType → Lists cost center values**

## **4. Module:Scrap – 551 Movement Transaction**

**1. Frontend Details**

**Description: This module allows users to raise, track, and download SAP 551 movement transactions (Stock posting for cost center consumption).  
 Users can either:**

* **Upload transactions via an Excel file.**
* **Or submit requests manually through a form interface.**

**Transactions are routed for approval, and the approval status can be viewed using the Doc\_ID.**

* **Route path: http://localhost:3001/home/scrap551**
* **Component path: src/dashboard/Scrap551.jsx**

**API Service File:**

* **src/controller/Movement551apiservice.js**
  + **API Functions Used** 
    - **Movement551(data) → POST /Movement551/File**
    - **Movement551Reupload() → POST /Movement551/FileReupload?Doc\_ID=<id>**
    - **getdetails(user) → GET /Movement551/get\_details?userid=<user>**
    - **Edit551Record(data) → POST /Movement551/Edit202Record**
    - **getresubmit(data)→ POST /Movement551/HandleResubmitAction551**
    - **getCancel(data)→ POST /Movement551/HandleCancelAction551?Doc\_ID**
    - **get551ApprovalView(id) → GET /Movement551/View551ApprovalStatus?Doc\_ID=<id>**
    - **getTransactionData()→ GET /Movement551/download\_data?From=...&to=...**
    - **DownloadAllExcel(id)→ GET /Movement551/download\_data?Doc\_ID=<id>**
    - **getPlants() → GET /Movement551/Get\_Plants**
    - **getMaterial()→ GET /Movement551/Get\_Material**
    - **getSLoc()→ GET /Movement551/Get\_StorageLocation**
    - **getMovement()→ GET /Movement551/Get\_MovementType**
    - **getReasonForMovement()→ GET /Movement551/Get\_ReasonForMovementType**
    - **getCostCenter()→ GET /Movement551/Get\_CostCenter**
    - **getView()→ GET /Movement551/get\_View**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Movement551.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Material**
  + **Mst\_Material\_Type**
  + **Mst\_Storage\_Location**
  + **Mst\_Valuation\_Type**
  + **Mst\_Movement\_List\_Item**
  + **Mst\_Movement\_Type**
  + **Mst\_User**
  + **Mst\_CostCenter**

**Purpose → Raised Request and Get Approval for**

**SAP – 551 Movement Transaction.**

* **Procedures Used:**
* **InsertTrn551Movt→ Handles insertion of new 551 movement transactions**
* **Edit551Record→ Updates a rejected or draft 551 record**
* **UploadTrn551SapTransferRecord → Handles bulk upload via Excel for 551 movement**
* **GetExistingTrn551Movt → Retrieves filtered 551 movement records for download(Excel export)**
* **GetExistingDocIDTrn551Movt → Downloads data of specific Doc\_ID (Excel export)**
* **ResubmitApproval → Resubmits a rejected approval request**
* **CancelApproval551 → Cancels an approval request**
* **View\_Stock551Approval\_Status → Gets approval status for given Doc\_ID**
* **GetTrnSap551Movt → Fetches detailed 551 movement record list**
* **GetActiveStorageLocation → Fetches active storage location list**
* **GetActiveMaterial → Returns all active materials**
* **GetActiveValuationType → Lists valuation types (dropdown)**
* **GetActivePlants → Returns available plants**
* **GetActiveMovementType → Lists movement types (551)**
* **Get551ReasonForMovement → Lists valid reasons for 551 transaction**
* **GetActiveCostCentertType → Lists cost center values**

## **5. Module:Location – 311 Movement Transaction**

**1. Frontend Details**

**Description: This module allows users to initiate, resubmit, track, and download SAP 311 Movement Transactions, typically used for stock transfer between storage locations within the same plant.  
 Users can either:**

* **Upload transactions via an Excel file.**
* **Or submit requests manually through a form interface.**

**Transactions are routed for approval, and the approval status can be viewed using the Doc\_ID.**

* **Route path: http://localhost:3001/home/Location**
* **Component path: src/dashboard/Location.jsx**

**API Service File:**

* **src/controller/Movement311apiservice.js**
  + **API Functions Used** 
    - **Movement311(data) → POST /Movement551/File**
    - **Movement311Reupload() → POST /Movement311/FileReupload?Doc\_ID=<id>**
    - **getdetails(user) → GET /Movement311/get\_details?userid=<user>**
    - **Edit311Record(data) → POST /Movement311/Edit311Record**
    - **getresubmit(data)→ POST /Movement311/HandleResubmitAction311**
    - **getCancel(data)→ POST /Movement311/HandleCancelAction311?Doc\_ID**
    - **get311ApprovalView(id) → GET /Movement311/View551ApprovalStatus?Doc\_ID=<id>**
    - **getTransactionData()→ GET /Movement311/getTransactionData?From=<>&to=<>**
    - **DownloadAllExcel(id)→ GET /Movement311/download\_data?Doc\_ID=<id>**
    - **getPlants() → GET /Movement311/Get\_Plants**
    - **getMaterial()→ GET /Movement311/Get\_Material**
    - **getSLoc()→ GET /Movement311/Get\_StorageLocation**
    - **getMovement()→ GET /Movement311/Get\_MovementType**
    - **getReasonForMovement()→ GET /Movement311/Get\_ReasonForMovementType**
    - **getCostCenter()→ GET /Movement311/Get\_CostCenter**
    - **getView()→ GET /Movement311/get\_View**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Movement311.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Material**
  + **Mst\_Storage\_Location**
  + **Mst\_Valuation\_Type**
  + **Mst\_Movement\_Type**
  + **Mst\_User**
  + **Mst\_CostCenter**

**Purpose → Raised Request and Get Approval for**

**SAP – 311 Movement Transaction.**

* **Procedures Used:**
* **InsertTrn3111Movt→ Handles insertion of new 311 movement transactions**
* **Edit311Record→ Updates a rejected or draft 311 record**
* **UploadTrn311SapTransferRecord → Handles bulk upload via Excel for 311 movement**
* **GetExistingTrn311Movt → Retrieves filtered 311 movement records for download(Excel export)**
* **GetExistingDocIDTrn311Movt → Downloads data of specific Doc\_ID (Excel export)**
* **ResubmitApproval → Resubmits a rejected approval request**
* **CancelApproval311 → Cancels an approval request**
* **View\_Stock311Approval\_Status → Gets approval status for given Doc\_ID**
* **GetTrnSap311Movt → Fetches detailed 3111 movement record list**
* **GetActiveStorageLocation → Fetches active storage location list**
* **GetActiveMaterial → Returns all active materials**
* **GetActiveValuationType → Lists valuation types (dropdown)**
* **GetActivePlants → Returns available plants**
* **GetActiveMovementType → Lists movement types (311)**
* **Get311ReasonForMovement → Lists valid reasons for 311 transaction**
* **GetActiveCostCentertType → Lists cost center values**

## **6. Module:Conversion Rs.1 Movement Transaction**

**1. Frontend Details**

**Description: This module allows users to upload, manage, resubmit, view, and track SAP Rs.1 Movement (Conversion) Transactions, primarily for internal stock valuation purposes across the same plant.   
 Users can either:**

* **Upload transactions via an Excel file.**
* **Or submit requests manually through a form interface.**

**Transactions are routed for approval, and the approval status can be viewed using the Doc\_ID.**

* **Route path: http://localhost:3001/home/ConversionRs1**
* **Component path: src/dashboard/Conversion.jsx**

**API Service File:**

* **src/controller/MovtCRs1apiservice.js**
  + **API Functions Used** 
    - **MovementConversion(data) → POST /Movement551/File**
    - **MovementRs1Reupload() → POST /Movement3Rs1/FileReupload?Doc\_ID=<id>**
    - **getdetails(user) → GET /MovementRs1/get\_details?userid=<user>**
    - **EditRecord(data) → POST /MovementRs1/EditRs1MovementRs1Record**
    - **getresubmit(data)→ POST /MovementRs1/HandleResubmitActionRs1MovementRs1**
    - **getCancel(data)→ POST /MovementRs1/HandleCancelActionRs1MovementRs1**
    - **getRs1ApprovalView(id) → GET /MovementRs1/ViewRePgApprovalStatus?Doc\_ID=<id>**
    - **getTransactionData(from, to)→ GET /MovementRs1/getTransactionData?From=<from>&to=<to>**
    - **DownloadAllExcel(id)→ GET /MovementRs1/download\_data?Doc\_ID=<id>**
    - **getPlants() → GET /MovementRs1/Get\_Plants**
    - **getMaterial()→ GET /MovementRs1/Get\_Material**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Movement311.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Material**
  + **Mst\_User**

**Purpose → Raised Request and Get Approval for**

**SAP – Rs.1 Movement Transaction.**

* **Procedures Used:**
* **UploadConversionRs1Record → Uploads valid Rs.1 transaction data from Excel (bulk insert)**
* **GetTrnConversionRs1Movt → Fetches Rs.1 movement transactions by Created\_By**
* **GetExistingConversionRS1 → Gets transactions filtered by FromDate and ToDate**
* **View\_ConversionRs1Approval\_Status → Returns approval status for a given Doc\_ID**
* **GetExistingDocIDTrnRs1MovementRs1Movt → Exports transaction rows for a Doc\_ID**
* **EditRs1MovementRs1Record → Edits a specific Rs.1 movement record**
* **ResubmitApproval → Handles re-submission of transactions**
* **CancelApprovalRs1MovementRs1 → Cancels a document submission**
* **GetTrnSap311Movt → Fetches detailed 3111 movement record list**
* **GetActivePlants → Returns list of valid plants**
* **GetActiveMaterial → Returns all active materials**
* **GetActiveMaterial → Returns list of valid materials**

**7. Module: Inward of Old Invoice Transaction**

**1.Frontend Details(Purchase)**

**Description: Add, view,Edit,Resubmit**

**Route path:**[**http://localhost:3001/home/Inward**](http://localhost:3001/home/Inward)

* **Component path: src/Dashboard/Purchase.jsx**

**API Service File:**

* **src/controller/Inwardtransactionapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsPurchase() → GET**

**/Inward/get\_Purchasedetails?UserID=${UserID}**

* + - **getAdd(data) → POST**

**/Inward/Get\_AddPurchase**

* + - **updateInwardInvoicePurchase(data) → PUT**

**/Inward/update\_Purchase**

* + - **getPurchaseData(data) → GET /Inward/downloadPurchase?From=${from}&to=${to}&UserID=${UserID}**
    - **Resubmit (data) → POST**

**/Inward/resubmit**

* + - **getVendor(data) → GET**

**/Inward/Get\_Vendor**

* + - **getMaterial(data) → GET /Inward/Get\_Material**

**2.Backend Details**

**Backend File: server/routes/Inward.Router.js**

* **SQL Tables: Trn\_Inward\_Old\_Invoice**
* **SQL Fetch Other Table: Mst\_Material,Mst\_vendor**

**Purpose → Stores Inward Transaction data**

* **Procedures Used:**

**1.GetPurchaseDetails → Fetch all Records**

**2.GetMaterialForInward→Fetch Active Material Records**

**3.GetVendorforInwards→Fetch Active Vendor Records**

**4.InsertTrnInwardOldInvoicePurchase→Insert Purchase Records**

**5.UpdateTrnInwardOldInvoicePurchase→Updated Existing Record in Purchase**

**6.GetInwardInvoicesByDateForPurchase→Fetch Purchase data by date Records**

**7. ResubmitRejectedInward→Update Records status**

**1.Frontend Details(Service)**

**Description: Add, view,Edit,Resubmit**

**Route path:**[**http://localhost:3001/home/Inward**](http://localhost:3001/home/Inward)

* **Component path: src/Dashboard/Service.jsx**

**API Service File:**

**src/controller/Inwardtransactionapiservice.jsx**

* + **API Functions Used** 
    - **getdetailsService() → GET**

**/Inward/get\_Servicedetails?UserID=${UserID}**

* + - **getAddService(data) → POST**

**/Inward/Get\_AddService**

* + - **updateInwardInvoiceService(data) → PUT**

**/Inward/update\_Service**

* + - **getServiceData(data) → GET /Inward/downloadService?From=${from}&to=${to}&UserID=${UserID}**
    - **Resubmit (data) → POST**

**/Inward/resubmit**

* + - **getVendor(data) → GET**

**/Inward/Get\_Vendor**

**2.Backend Details**

**Backend File: server/routes/Inward.Router.js**

* **SQL Tables: Trn\_Inward\_Old\_Invoice**
* **SQL Fetch Other Table: Mst\_vendor**

**Purpose → Stores Inward Transaction data**

* **Procedures Used:**

**1.GetServiceDetails → Fetch all Records**

**2.GetVendorforInwards→Fetch Active Vendor Records**

**3.InsertTrnInwardOldInvoiceService→Insert Service Records**

**4.UpdateTrnInwardOldInvoiceService→Updated Existing Record in Service**

**5.GetInwardInvoicesByDateForService→Fetch Service data by date Records**

**6.ResubmitRejectedInward→Update Records status**

**8. Module: Emergency Procurement**

1. **Frontend Details**

**Description: Add, view,Edit,Resubmit**

**Route path:****http://localhost:3001/home/Emergency**

* **Component path: src/Dashboard/Emergency.jsx**

**API Service File:**

* **src/controller/EmergencyProcurementapiservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET**

**/Emergency/get\_details?UserID=${UserID}**

* + - **getAdd(data) → POST**

**/Emergency/Get\_Add**

* + - **getUpdates(data) → PUT**

**/Emergency/get\_Updates**

* + - **getEmergencyData(data) → GET /Emergency/downloadEmergency?From=${from}&to=${to}&UserID=${UserID}**
    - **Resubmit (data) → POST**

**/Inward/resubmit**

* + - **getVendor(data) → GET**

**/Emergency/Get\_Vendor**

1. **Backend Details**

**Backend File: server/routes/EmergencyProcurement.Router.js**

* **SQL Tables: Trn\_Inward\_Old\_Invoice**
* **SQL Fetch Other Table:Mst\_vendor**

**Purpose → Stores Inward Transaction data**

* **Procedures Used:**

**1.GetEmergencyProcurment → Fetch all Records**

**2.GetVendorforInwards→Fetch Active Vendor Records**

**3.InsertTrnEmergencyProcurment→Insert Emergency Procurement Records**

**4.UpdateTrnEmergencyProcurement→Updated Existing Record in Emergency Procurement**

**5.GetInwardInvoicesByDateForEmergency→Fetch Emergency Procurement data by date 6..ResubmitRejectedInward→Update Records status**

**—--------------------------------------------------**

* **Approval**

**1. Module:309 Approval**

**1. Frontend Details**

**Description: The 309 Approval module is responsible for managing the approval workflow of SAP 309 Movement transactions. It enables approvers to review, approve, reject, or query transactions that are routed to them based on Plant and Approval Level (Role).   
Users can:**

* **Filter transactions based on Plant and Role**
* **View all pending transactions**
* **Take action (approve, reject, query)**
* **Download approval data (summary or detailed)**
* **Track document-level approval history via Doc\_ID**
* **Route path: http://localhost:3001/home/Approval\_309**
* **Component path: src/dashboard/Approval.jsx**

**API Service File:**

* **src/controller/Approvalapiservice.js**
  + **API Functions Used** 
    - **getdetails(Plant, role, Role) → GET /Approval/get\_details?Plant=<>&role=<>&Role=<>**
    - **getApprovalView(docId) → GET /Approval/get\_ViewButton?Doc\_ID=<id>**
    - **get309ApprovalView(docId) → GET /Approval/View309ApprovalStatus?Doc\_ID=<id>**
    - **HandleApprovalAction(data) → POST /Approval/HandleApprovalAction**
    - **resubmitAction(docId) → POST /Approval201/HandleResubmitAction**
    - **getPlants()→ GET /Approval/Get\_Plants**
    - **getRole()→ GET /Approval/Get\_Role**
    - **get\_detailsApproval() → GET /Approval/get\_detailsApproval?Plant=<>&role=<>&Role=<>**
    - **DownloadAllExcel(docId) → GET /Approval/download\_data?Doc\_ID=<id>**
    - **DownloadDocumentAllExcel(docId)→ GET /Approval/download\_AllRow\_Docdata?Doc\_ID=<id>**
    - **fetchDocumentExcelData(docId) → GET /Approval/download\_AllRow\_Docdata?Doc\_ID=<id>**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Approval.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Role**
  + **Mst\_User**
* **Procedures Used:**
* **ApprovalListView → Fetches pending transactions based on Plant and Role**
* **ApprovalViewButton → Retrieves grouped data for a specific document**
* **View309Approval\_Status → Returns full approval trail for a given Doc\_ID**
* **Trn309\_ApprovalWorkflow → Handles approve/reject/query actions based on user input**
* **GetActivePlants → Returns all valid plants for approval filtering**
* **GetActiveRole → Returns list of roles involved in the approval process**
* **GetAllExistingTrn309Movt → Returns all records for a given Doc\_ID (used for detailed Excel export)**

**2. Module:201 Approval**

**1. Frontend Details**

**Description: The 201 Approval module is responsible for managing the approval workflow for SAP 201 Movement Transactions. It enables approvers to review, approve, reject, or query transactions that are routed to them based on Plant and Approval Level (Role).   
Users can:**

* **Filter transactions based on Plant and Role**
* **View all pending transactions**
* **Take action (approve, reject, query)**
* **Download approval data (summary or detailed)**
* **Track document-level approval history via Doc\_ID**
* **Route path: http://localhost:3001/home/Approval\_201**
* **Component path: src/dashboard/Approval\_201.jsx**

**API Service File:**

* **src/controller/Approval201apiservice.js**
  + **API Functions Used** 
    - **getdetails(Plant, role, Role) → GET /Approval201/get\_details?Plant=<>&role=<>&Role=<>**
    - **getApprovalView(docId) → GET /Approval201/get\_ViewButton?Doc\_ID=<id>**
    - **get201ApprovalView(docId) → GET /Approval201/View201ApprovalStatus?Doc\_ID=<id>**
    - **resubmitAction(docId) → POST /Approval201/HandleResubmitAction**
    - **HandleApprovalAction(data) → POST /Approval201/HandleApprovalAction**
    - **getPlants()→ GET /Approval201/Get\_Plants**
    - **getRole()→ GET /Approval201/Get\_Role**
    - **get\_detailsApproval() → GET /Approval201/get\_detailsApproval?Plant=<>&role=<>&Role=<>**
    - **DownloadAllExcel(docId) → GET /Approval201/download\_data?Doc\_ID=<id>**
    - **DownloadDocumentAllExcel(docId)→ GET /Approval201/download\_AllRow\_Docdata?Doc\_ID=<id>**
    - **HandleCancelAction() → POST /Approval201/HandleCancelAction**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Approval201.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Role**
  + **Mst\_User**
* **Procedures Used:**
* **ApprovalListView201 → Fetches pending 201 transactions based on Plant and Role**
* **ApprovalViewButton → Retrieves grouped data for a specific document**
* **View201Approval\_Status → Returns full approval trail for a given Doc\_ID**
* **Trn201\_ApprovalWorkflow → Handles approval actions (approve/reject/query)**
* **GetActivePlants → Returns all valid plants for approval filtering**
* **GetActiveRole → Returns list of roles involved in the approval process**
* **GetView\_201ApprovalStatus\_ByID → Returns view/status for a user-specific approval document**
* **ResubmitApproval → Returns view/status for a user-specific approval document**
* **GetExistingDocIDApproval201Movt → Returns all document rows for Excel export based on Doc\_ID**

**3. Module:202 Approval**

**1. Frontend Details**

**Description: The 202 Approval module is responsible for managing the approval workflow for SAP 202 Movement Transactions. It enables approvers to review, approve, reject, or query transactions that are routed to them based on Plant and Approval Level (Role).   
Users can:**

* **Filter transactions based on Plant and Role**
* **View all pending transactions**
* **Take action (approve, reject, query)**
* **Download approval data (summary or detailed)**
* **Track document-level approval history via Doc\_ID**
* **Route path: http://localhost:3001/home/Approval\_202**
* **Component path: src/dashboard/Approval\_202.jsx**

**API Service File:**

* **src/controller/Approval202apiservice.js**
  + **API Functions Used** 
    - **getdetails(Plant, role, Role) → GET /Approval202/get\_details?Plant=<>&role=<>&Role=<>**
    - **getApprovalView(docId) → GET /Approval202/get\_ViewButton?Doc\_ID=<id>**
    - **get202ApprovalView(docId) → GET /Approval202/View201ApprovalStatus?Doc\_ID=<id>**
    - **resubmitAction(docId) → POST /Approval202/HandleResubmitAction**
    - **HandleApprovalAction(data) → POST /Approval202/HandleApprovalAction**
    - **getPlants()→ GET /Approval202/Get\_Plants**
    - **getRole()→ GET /Approval202/Get\_Role**
    - **get\_detailsApproval() → GET /Approval202/get\_detailsApproval?Plant=<>&role=<>&Role=<>**
    - **DownloadAllExcel(docId) → GET /Approval202/download\_data?Doc\_ID=<id>**
    - **DownloadDocumentAllExcel(docId)→ GET /Approval202/download\_AllRow\_Docdata?Doc\_ID=<id>**
    - **HandleCancelAction() → POST /Approval202/HandleCancelAction**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Approval202.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Role**
  + **Mst\_User**
* **Procedures Used:**
* **ApprovalListView202 → Fetches pending 202 transactions based on Plant and Role**
* **ApprovalViewButton → Retrieves grouped data for a specific document**
* **View202Approval\_Status → Returns full approval trail for a given Doc\_ID**
* **Trn202\_ApprovalWorkflow → Handles approval actions (approve/reject/query)**
* **GetActivePlants → Returns all valid plants for approval filtering**
* **GetActiveRole → Returns list of roles involved in the approval process**
* **GetView\_202ApprovalStatus\_ByID → Returns view/status for a user-specific approval document**
* **ResubmitApproval → Returns view/status for a user-specific approval document**
* **GetExistingDocIDApproval202Movt → Returns all document rows for Excel export based on Doc\_ID**

**4. Module:551 Approval**

**1. Frontend Details**

**Description: The 551 Approval module is responsible for managing the approval workflow for SAP 551 Movement Transactions. It enables approvers to review, approve, reject, or query transactions that are routed to them based on Plant and Approval Level (Role).   
Users can:**

* **Filter transactions based on Plant and Role**
* **View all pending transactions**
* **Take action (approve, reject, query)**
* **Download approval data (summary or detailed)**
* **Track document-level approval history via Doc\_ID**
* **Route path: http://localhost:3001/home/Approval\_551**
* **Component path: src/dashboard/Approval\_202.jsx**

**API Service File:**

* **src/controller/Approval551apiservice.js**
  + **API Functions Used** 
    - **getdetails(Plant, role, Role) → GET /Approval551get\_details?Plant=<>&role=<>&Role=<>**
    - **getApprovalView(docId) → GET /Approval551/get\_ViewButton?Doc\_ID=<id>**
    - **get551ApprovalView(docId) → GET /Approval551/View551ApprovalStatus?Doc\_ID=<id>**
    - **resubmitAction(docId) → POST /Approval551/HandleResubmitAction**
    - **HandleApprovalAction(data) → POST /Approval5511/HandleApprovalAction**
    - **getPlants()→ GET /Approval551/Get\_Plants**
    - **getRole()→ GET /Approval551/Get\_Role**
    - **get\_detailsApproval() → GET /Approval551/get\_detailsApproval?Plant=<>&role=<>&Role=<>**
    - **DownloadAllExcel(docId) → GET /Approval551/download\_data?Doc\_ID=<id>**
    - **DownloadDocumentAllExcel(docId)→ GET /Approval551/download\_AllRow\_Docdata?Doc\_ID=<id>**
    - **HandleCancelAction() → POST /Approval551/HandleCancelAction**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Approval551.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Role**
  + **Mst\_User**
* **Procedures Used:**
* **ApprovalListView551 → Fetches pending 551 transactions based on Plant and Role**
* **ApprovalViewButton → Retrieves grouped data for a specific document**
* **View551Approval\_Status → Returns full approval trail for a given Doc\_ID**
* **Trn551\_ApprovalWorkflow → Handles approval actions (approve/reject/query)**
* **GetActivePlants → Returns all valid plants for approval filtering**
* **GetActiveRole → Returns list of roles involved in the approval process**
* **GetView\_551ApprovalStatus\_ByID → Returns view/status for a user-specific approval document**
* **ResubmitApproval → Returns view/status for a user-specific approval document**
* **GetExistingDocIDApproval551Movt → Returns all document rows for Excel export based on Doc\_ID**

**5. Module:311 Approval**

**1. Frontend Details**

**Description: The 313 Approval module is responsible for managing the approval workflow for SAP 311 Movement Transactions. It enables approvers to review, approve, reject, or query transactions that are routed to them based on Plant and Approval Level (Role).   
Users can:**

* **Filter transactions based on Plant and Role**
* **View all pending transactions**
* **Take action (approve, reject, query)**
* **Download approval data (summary or detailed)**
* **Track document-level approval history via Doc\_ID**
* **Route path: http://localhost:3001/home/Approval\_311**
* **Component path: src/dashboard/Approval\_311.jsx**

**API Service File:**

* **src/controller/Approval311apiservice.js**
  + **API Functions Used** 
    - **getdetails(Plant, role, Role) → GET /Approval311/get\_details?Plant=<>&role=<>&Role=<>**
    - **getApprovalView(docId) → GET /Approval311/get\_ViewButton?Doc\_ID=<id>**
    - **get311ApprovalView(docId) → GET /Approval311/View311ApprovalStatus?Doc\_ID=<id>**
    - **resubmitAction(docId) → POST /Approval311/HandleResubmitAction**
    - **HandleApprovalAction(data) → POST /Approval311/HandleApprovalAction**
    - **getPlants()→ GET /Approval311/Get\_Plants**
    - **getRole()→ GET /Approval311/Get\_Role**
    - **get\_detailsApproval() → GET /Approval311/get\_detailsApproval?Plant=<>&role=<>&Role=<>**
    - **DownloadAllExcel(docId) → GET /Approval311/download\_data?Doc\_ID=<id>**
    - **DownloadDocumentAllExcel(docId)→ GET /Approval311/download\_AllRow\_Docdata?Doc\_ID=<id>**
    - **HandleCancelAction() → POST /Approval311/HandleCancelAction**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/Approval.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Role**
  + **Mst\_User**
* **Procedures Used:**
* **ApprovalListView311 → Fetches pending 311 transactions based on Plant and Role**
* **ApprovalViewButton → Retrieves grouped data for a specific document**
* **View311Approval\_Status → Returns full approval trail for a given Doc\_ID**
* **Trn311\_ApprovalWorkflow → Handles approval actions (approve/reject/query)**
* **GetActivePlants → Returns all valid plants for approval filtering**
* **GetActiveRole → Returns list of roles involved in the approval process**
* **GetView\_311ApprovalStatus\_ByID → Returns view/status for a user-specific approval document**
* **ResubmitApproval → Returns view/status for a user-specific approval document**
* **GetExistingDocIDApproval311Movt → Returns all document rows for Excel export based on Doc\_ID**

**6. Module: Conversion - Rs1 Approval**

**1. Frontend Details**

**Description: This module is used to manage, view, approve, reject, query, resubmit, or cancel RS1 conversion requests. It includes multiple API endpoints for interacting with the database and downloading related Excel data.  
Users can:**

* **Filter transactions based on Plant and Role**
* **View all pending transactions**
* **Take action (approve, reject, query)**
* **Download approval data (summary or detailed)**
* **Track document-level approval history via Doc\_ID**
* **Route path: http://localhost:3001/home/Approval\_Rs1**
* **Component path: src/dashboard/Approval\_202.jsx**

**API Service File:**

* **src/controller/ApprovalRs1apiservice.js**
  + **API Functions Used** 
    - **getdetails(Plant, role, Role) → GET /ApprovalRs1/get\_ApprovalPgdetails?Plant=<>&role=<>**
    - **getApprovalView(docId) → GET /ApprovalRs1/get\_ViewButton?Doc\_ID=<id>**
    - **getRs1ApprovalView(docId) → GET /ApprovalRs1/ViewRs1ApprovalStatus?Doc\_ID=<id>**
    - **resubmitAction(docId) → POST /ApprovalRs1/HandleResubmitAction**
    - **HandleApprovalAction(data) → POST /ApprovalRs1/HandleApprovalAction**
    - **getPlants()→ GET /ApprovalRs1/Get\_Plants**
    - **getRole()→ GET /ApprovalRs1/Get\_Role**
    - **get\_detailsApproval(userId) → GET /ApprovalRs1/get\_detailsApproval?userid=<userId>**
    - **DownloadAllExcel(docId) → GET /ApprovalRs1/download\_data?Doc\_ID=<id>**
    - **HandleCancelAction() → POST /Approval202/HandleCancelAction**

**2. Backend Details**

**DataBase: Sap\_Approval**

**Backend File: server/routes/ApprovalRs1.Router.js**

* **SQL Tables: Trn\_SapTransfer\_Records**
* **SQL Fetch Other Table:** 
  + **Mst\_Plant**
  + **Mst\_Role**
  + **Mst\_User**
* **Procedures Used:**
* **ApprovalListViewConversionRs1 → Fetches RS1 transactions filtered by Plant and Role**
* **ViewRs1Approval\_Status → Fetches full approval status/trail for a document**
* **ApprovalViewButton → Retrieves grouped data for a specific document**
* **GetActivePlants → Returns all available plants**
* **TrnConversionRs1\_ApprovalWorkflow → Handles approval actions (approve/reject/query)**
* **GetActiveRole → Returns all roles**
* **GetView\_Rs1ApprovalStatus\_ByID → Returns approval view for logged-in user**
* **ResubmitApproval → Handles document resubmission after query/rejection**
* **CancelApproval → Cancels an existing RS1 approval**
* **GetExistingDocIDTrnRs1Movt → Returns document data rows for Excel export**

**7. Module:Inward of Old Invoice Approval**

**1**.**Frontend Details(Purchase)**

**Description: The Inward of Old Invoice Approval(Purchase)**

**module is responsible for managing the approval workflow for SAP Inward Of Old Invoice. It enables approvers to view, approve, Reject transactions that are routed to them based on Plant and Approval Level (Role).**

**Route path:http://localhost:3001/home/InwardApproval**

* **Component path: src/Approval/InwardApproval.jsx**

**API Service File: Purchase**

* **src/controller/InwardApprovalservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET /InwardApproval/get\_details?UserID=${UserID}&RoleID=${RoleID}&PlantID=${PlantID}&EmployeeID=${EmployeeID}**
    - **getUpdates(data) → PUT**

**/InwardApproval/Get\_Updates**

* + - **getRejected(data) → GET /InwardApproval/Rejected**

* + - **getEmployee(data) → GET /InwardApproval/Get\_Employee**
    - **UpdateL2Approver(data)→ PUT**

**/InwardApproval/UpdateL2Approver**

**2.Backend Details**

**Backend File: server/routes/InwardApproval.Router.js**

* **SQL Tables: Trn\_Inward\_Old\_Invoice**

**Purpose → Stores Inward Of Old Invoice(Purchase)Transaction data**

* **Procedures Used:**

**1.GetPendingApprovalsForInwards → Fetch Pending Records**

**2.RejectApprovalStatusForInwards→Update the Status Emergency Procurement Records**

**3.UpdateApprovalStatusForInwards→Update the Status of Purchase Records**

**4.GetEmployeeDetails→ Fetch Employee Records**

**5.UpdateL2Approver→Update the Status of Purchase Records**

**1**.**Frontend Details(Service)**

**Description: The Inward of Old Invoice Approval(Service)**

**module is responsible for managing the approval workflow for SAP Inward Of Old Invoice. It enables approvers to view, approve, Reject transactions that are routed to them based on Plant and Approval Level (Role).**

**Route path:http://localhost:3001/home/InwardApproval**

* **Component path: src/Approval/InwardApproval.jsx**

**API Service File: Service**

* **src/controller/InwardApprovalservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET**

**/InwardApproval/get\_details?UserID=${UserID}&RoleID=${RoleID}&PlantID=${PlantID}&EmployeeID=${EmployeeID}**

* + - **getUpdates(data) → PUT**

**/InwardApproval/Get\_Updates**

* + - **getRejected(data) → GET /InwardApproval/Rejected**

**2.Backend Details**

**Backend File: server/routes/InwardApproval.Router.js**

* **SQL Tables: Trn\_Inward\_Old\_Invoice**

**Purpose → Stores Inward Of Old Invoice(Service)Transaction data**

* **Procedures Used:**

**1.GetPendingApprovalsForInwards → Fetch Pending Records**

**2.RejectApprovalStatusForInwards→Update the Status Emergency Procurement Records**

**3.UpdateApprovalStatusForInwards→Update the Status of Purchase Records**

**8. Module:Emergency Procurement Approval**

**1. Frontend Details**

**Description: The Emergency Procurement Approval module is responsible for managing the approval workflow for SAP Emergency Procurement. It enables approvers to view, approve, Reject transactions that are routed to them based on Plant and Approval Level (Role).**

**Route path:http://localhost:3001/home/EmergencyApproval**

* **Component path: src/Approval/EmergencyProcurementApproval.jsx**

**API Service File:**

* **src/controller/InwardApprovalservice.jsx** 
  + **API Functions Used** 
    - **getdetailsEmergency() → GET /InwardApproval/get\_details\_emergency?UserID=${UserID}&RoleID=${RoleID}&PlantID=${PlantID}&EmployeeID=${EmployeeID}**
* **EmergencyAprrove(data) → PUT**

**/InwardApproval/EmergencyAprrove**

* **getRejected(data) → GET /InwardApproval/Rejected**

1. **Backend Details**

**Backend File: server/routes/InwardApproval.Router.js**

* **SQL Tables: Trn\_Inward\_Old\_Invoice**

**Purpose → Stores Emergency Procurement Transaction data**

* **Procedures Used:**

**1.GetPendingApprovalsForEmergency → Fetch Pending Records**

**2.RejectApprovalStatusForInwards→Update the Status Emergency Procurement Records**

**3.UpdateApprovalStatusForInwards→Updated Existing Record in Emergency Procurement**

**4.Get\_Pending\_PO→ Fetch Pending PO Records**

**—--------------------------------------------------**

* **StoreDashboard**

**Module:Store Dashboard**

1. **Frontend Details**

**Description:View**

**Routepath:http://localhost:3001/home/StoreDashboard**

**Component path: src/Store Dashboard/StoreDashboard.jsx**

**API Service File:**

* **src/controller/StoreDashboardapiservice.jsx**
  + **API Functions Used** 
    - **getActiveStores() → GET**

**/StoreDashboard/get\_Store1Opendetails**

**Component path: src/Store Dashboard/Store2Open.jsx**

**API Service File:**

* **src/controller/StoreDashboardapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsStore1Open() → GET**

**/StoreDashboard/get\_Store1Opendetails**

**Component path: src/Store Dashboard/Store1Open.jsx**

**API Service File:**

* **src/controller/StoreDashboardapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsStore1Open() → GET**

**/StoreDashboard/get\_Store1Opendetails**

**Component path: src/Store Dashboard/Store3Open.jsx**

**API Service File:**

* **src/controller/StoreDashboardapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsStore1Open() → GET**

**/StoreDashboard/get\_Store1Opendetails**

**Component path: src/Store Dashboard/Store1Closed.jsx**

**API Service File:**

* **src/controller/StoreDashboardapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsStoreClosed() → GET**

**/StoreDashboard/get\_detailsStoreClosed**

* + - **getdetailsExcelDownload() → GET**

**/StoreDashboard/get\_detailsStoreExcelDownload**

* + - **getdetailsStoreClosedByDate() → GET**

**/StoreDashboard/StoreCloseddate**

**Component path: src/Store Dashboard/Store2Closed.jsx**

**API Service File:**

* **src/controller/StoreDashboardapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsStoreClosed() → GET**

**/StoreDashboard/get\_detailsStoreClosed**

* + - **getdetailsExcelDownload() → GET**

**/StoreDashboard/get\_detailsStoreExcelDownload**

* + - **getdetailsStoreClosedByDate() → GET**

**/StoreDashboard/StoreCloseddate**

**Component path: src/Store Dashboard/Store3Closed.jsx**

**API Service File:**

* **src/controller/StoreDashboardapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsStoreClosed() → GET**

**/StoreDashboard/get\_detailsStoreClosed**

* + - **getdetailsExcelDownload() → GET**

**/StoreDashboard/get\_detailsStoreExcelDownload**

* + - **getdetailsStoreClosedByDate() → GET**

**/StoreDashboard/StoreCloseddate**

1. **Backend Details**

**Backend File: server/routes/SapMaster.Router.js**

* **SQL Tables: Trn\_prod\_order\_status**
* **SQL Fetch Other Table:TrnSap\_Order\_creation,TrnSap\_Order\_Closed**

**Purpose → Stores Production Order Data**

**Procedures Used:**

**1.GetProductionOrderStatus → Fetch Order details for OpenStores records**

**2.GetProductionClosedStatus→ Fetch Order details for ClosedStores records**

**3.GetProductionClosedStatusExcel→Fetch Store data to Download Records**

**4.GetProductionClosedExport→Fetch Closed Store Records**

**5.Sync\_Prod\_Order\_Status→Fetch and Insert New And Closed Order details**

**6.GetActiveStoresByPlant→ Fetch ActiveStore details**

**2. Module:StoreUrl**

1. **Frontend Details**

**Description:View**

**Routepath:**

**http://localhost:3001/Store/PlantCode/StorageCode**

**Component path: src/Store Dashboard/Store1Url.jsx**

**API Service File:**

* **src/StoreUrl/StoreUrlapiservice.jsx** 
  + **API Functions Used** 
    - **getdetailsStoreOpen() → GET**

**/StoreUrl/get\_StoreOpendetails**

* + - **getActiveStores() → GET**

**/StoreUrl/getActiveStores**

1. **Backend Details**

**Backend File: server/routes/SapMaster.Router.js**

* **SQL Tables: Trn\_prod\_order\_status**
* **SQL Fetch Other Table:TrnSap\_Order\_creation,TrnSap\_Order\_Closed**

**Purpose → Stores Production Order Data**

**Procedures Used:**

**1.GetProductionOrderStatusForUrl → Fetch Order details for OpenStores records**

**2.Sync\_Prod\_Order\_Status→Fetch and Insert New And Closed Order details**

**3.GetActiveStoresByPlantStorage→ Fetch ActiveStore details**

**—------------------------------------------------------**

**7. Module:Sap Login Access**

1. **Frontend Details**

**Description: Add, View,Edit data**

**Route path:****http://localhost:3001/home/SAP**

* **Component path: src/Sap User Access/Sap.jsx**

**API Service File:**

* **src/controller/AdminMasterapiservice.jsx** 
  + **API Functions Used** 
    - **getdetails() → GET**

**/SapMaster/get\_details**

* + - **getAdd(data) → POST**
    - **/SapMaster/Get\_Add**
    - **getUpdate(data) → PUT**

**/SapMaster/get\_Updates**

* + - **getPlants(data)→ GET**

**/Master/Get\_Plants**

* + - **getCategory(data)→ GET**

**/SapMaster/Get\_Category**

1. **Backend Details**

**Database :DHRM\_23052025,Sap\_Approval**

**Backend File: server/routes/SapMaster.Router.js**

* **SQL Tables: Mst\_Sap\_Login**
* **SQL Fetch Other Database: DHRM\_23052025**
* **SQL Fetch Other Table:employees,ars\_dump\_data,trainee\_apln**

**Purpose → Stores Sap Login User Data**

* **Procedures Used:**

1. **GetSapDetails → Fetch all Sap records**
2. **InsertMST\_Sap→ Insert new Sap Login User**
3. **GetDHRMEmployee→Fetch Employee Records**
4. **GetDHRMTemporary→Fetch Contract Employee Records**
5. **UpdateSapMaster→Update Existing Sap Login User details**
6. **Sap\_UpdateLastPunch→ Fetch LastPunch details**
7. **GetSapdetailemail → Fetch Email id details**