DESIGNING A RESPONSIVE MASTER LIST COMPONENT WITH BASED ON ACCESS GROUPS IN PEGA CONSTELLATION

NAME OF AUTHOR: MADDI DINESH
UNDER GUIDANCE OF: NIRUP CHERLAKOLA

Environment Setup and Component Creation

To set up your development environment and create a custom widget component, you can refer to the first 5 pages of the document shared in the following LinkedIn post:

A guide to overriding the Pega static table

Problem Statement

We needed a flexible, reusable component to **view, edit, delete**, and **download** Master List data with structured hierarchy and access control:

- Data is structured as **Group → Region → Country**
- Based on the logged-in user's role, certain actions are shown or hidden
- The component connects to Pega's **Data Pages** using **DX API**
- It supports case opening directly from the table
- A download option is available for exporting the data

1.Interfaces Decleration

To work with the data, we created the following interfaces in TypeScript to define the structure:

```
export interface Group {
  GroupRMEmail: string;
  GroupName: string;
  GroupID: string;
  GroupRM: string;


  CINs: string;
  GPSClientTier: string;
}
export interface Region {
  RegionName: string;
  RegionRMEmail: string;
  RegionRM: string;
  pxObjClass: string;
}
export interface Country {
  CountryName: string;
  CountryRM: string;
  CountryRMEmail: string;
  ServicingEmail: string;
```

```
 pxObjClass: string;
}
export interface MasterDataItem {
  Group: Group;
  Regions: Region[];
  Countries: Country[];
  ActiveGroup?: Group;
  ActiveRegions?: Region[];
  ActiveCountries?: Country[];
  pyID: string;
  pxObjClass: string;
  pzInsKey: string;
  Status: string;
}
export interface MasterDataResponse {
  data: MasterDataItem[];
}
export interface AuthResponse {
  access_token: string;
  token_type: string;
  expires_in: number;
}
export interface ApiConfig {
  clientId: string;
  clientSecret: string;
  tokenEndpoint: string;
  username: string;
  password: string;
  apiEndpoint: string;
  grantType: string;
}
```

2. Fetching Data from Pega Using DX API

We used **Pega DX APIs** to fetch the data from **Data Pages** (D_AllCases and D_BulkDownload). We also used **OAuth 2.0** to get the access token before calling any API.

2.1 Authentication – Get Access Token

This function connects to Pega and fetches an access token using client credentials and username/password:

```
export const getAuthToken = async (): Promise<string> => {
  try {
    const response = await axios.post<AuthResponse>(
      apiConfig.tokenEndpoint,
      new URLSearchParams({
        grant_type: apiConfig.grantType,
        client_id: apiConfig.clientId, client_secret: apiConfig.clientSecret,
        username: apiConfig.username,
        password: apiConfig.password
      }),
      {
        headers: {
          'Content-Type': 'application/x-www-form-urlencoded'

        }
      }
    );
    return response.data.access_token;
  } catch (error) {
```

```
    throw new Error('Failed to obtain authentication token');
  }
};
```

2.2 Fetch Master List Data

After getting the token, we call the data view API D_AllCases to fetch the list of Groups, Regions, and Countries:

```
export const fetchMasterData = async (): Promise<MasterDataResponse> => {
  try {
    const token = await getAuthToken();
    const response = await axios.post<MasterDataResponse>(
      apiConfig.apiEndpoint,
      {}, // Empty body
      {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      }
    );
    return response.data;
  } catch (error) {
    throw new Error('Failed to fetch master data');
  }
};
```

2.3 Fetch Data for Excel Download

We used another Data Page D_BulkDownload to get the downloadable data. This also uses the same token.

```
export const fetchBulkDownloadData = async () => {
  try {
    const token = await getAuthToken();
    const { data } = await axios.post(
      'https://truviq02.pegalabs.io/prweb/api/application/v2/data_views/D_BulkDownload',
      {}, // Empty body
      {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      }
    );
    return data;
  } catch (error) {
    throw new Error('Failed to fetch bulk download data');
  }
};
```

3. Import the required packages and dependencies

Start by importing the necessary React hooks and Pega Cosmos components. Also, import utility functions from your api.ts and types.ts files.

If xlsx and axios are not already installed in your project, install them using the following command:
>npm install xlsx axios

Then, add the following imports to your component index.tsx file:

```
import React, { useEffect, useState, useMemo } from 'react';
import { Card, CardHeader, CardContent, Icon, Input, Button, withConfiguration } from '@pega/cosmos-react-core';

import * as XLSX from 'xlsx';

import * as pencilIcon from '@pega/cosmos-react-core/lib/components/Icon/icons/pencil.icon';

import { registerIcon } from '@pega/cosmos-react-core';

import { fetchMasterData, fetchBulkDownloadData } from './api';

import { StyledWrapper } from './styles';

import type { MasterDataItem, Group, Region, Country } from './types';
```

4.State and Setup

- **State Hooks**: The component uses useState to manage various parts of its state, including:
    - data: The main array of master list items.
    - loading: A boolean to show a loading spinner.
    - error: A string to display error messages.
    - isDownloading: Tracks the state of the bulk download.
    - operatorId: Stores the logged-in user's role to control permissions.
    - searchTerm: Holds the search query from the user.
    - expandedRows: A Set to track which rows in the hierarchy are open.
    - sortConfig: An object to manage sorting by a specific column.
- **useEffect Hooks**:
    - The first useEffect runs when the component loads. It registers the "pencil" icon and gets the current user's ID from the PCore object, which is a global Pega library.
    - The second useEffect calls the loadData function to fetch the data from Pega.

```
const [loading, setLoading] = useState(true);
const [error, setError] = useState<string | null>(null);
const [searchTerm, setSearchTerm] = useState('');
const [expandedRows, setExpandedRows] = useState<Set<string>>(new Set());
const [sortConfig, setSortConfig] = useState<{ key: string; direction: 'asc'|'desc'}|null>(null);
const [data, setData] = useState<MasterDataItem[]>([]);
const [isDownloading, setIsDownloading] = useState(false);
const [operatorId, setOperatorId] = useState<string>('');
```

5.Functionality and Helpers

- **handleBulkDownload()**: This function handles the download process. It gets the download data, defines the column headers, and uses the xlsx library to create an Excel file with a timestamped name.

```
const handleBulkDownload = async () => {
    try {
      setIsDownloading(true);
      const response = await fetchBulkDownloadData();
```

```
      // Define columns in specific order
      const columns = [
        'GroupID', 'GroupName', 'GPSClientTier', 'CINs', 'GroupRM', 'GroupRMEmail',
        'RegionName', 'RegionRM', 'RegionRMEmail',
        'CountryName', 'CountryRM', 'CountryRMEmail', 'ServicingMail'
      ];
      // Create worksheet
      const ws = XLSX.utils.json_to_sheet(response.data, {
        header: columns
      });
      // Add filters
      ws['!autofilter'] = { ref: `A1:${String.fromCharCode(65 + columns.length - 1)}1` };
      // Create workbook
      const wb = XLSX.utils.book_new();
      XLSX.utils.book_append_sheet(wb, ws, 'Master Groups');
      // Generate timestamp
      const timestamp = new Date().toISOString().replace(/[:.]/g, '-').slice(0, -5);
      const fileName = `Master_Groups_${timestamp}.xlsx`;
      // Trigger file save dialog
      XLSX.writeFile(wb, fileName, { bookType: 'xlsx', bookSST: false });
    } catch (err) {

    } finally {
      setIsDownloading(false);
    }
  };
```

- **handleCaseOpen():** This function is triggered when the edit icon is clicked. It uses Pega's getActionsApi to open a Pega case directly from the component. The getAssignmentId helper function determines the correct case to open based on the data item's status.

```
const getAssignmentId = (pyId: string, status: string): string => {
    const baseId = `ASSIGN-WORKLIST ABC-MASTERDATA-WORK ${pyId}`;
    if (status === 'PENDING APPROVAL') {
      return `${baseId}!PYCASCADINGGETAPPROVAL`;
    }
    return `${baseId}!Active_Flow`;
  };
const handleCaseOpen = (item: MasterDataItem) => {
  const PCore = (window as any).PCore;
    if (!PCore) return;

    const className = ABC-MasterData-Work-MasterDataManagement';
    const containerName = 'primary';
    const assignmentID = getAssignmentId(item.pyID, item.Status);

    const actions = getPConnect().getActionsApi();
    actions.openAssignment(assignmentID, className, { containerName });
  };
```

- **shouldShowEditIcon():** This simple function checks the operatorId and the row's Status to decide if the edit icon should be visible. This is a form of **access control**.

```
useEffect(() => {
    registerIcon(pencilIcon);
    const PCore = (window as any).PCore;
    if (PCore) {
      const currentOperator = PCore.getEnvironmentInfo().getOperatorIdentifier() ?? '';
      setOperatorId(currentOperator);
```

```
  }
}, []);

const shouldShowEditIcon = (status: string): boolean => {
  if (operatorId === 'reviewmanager') {
    return status === 'PENDING APPROVAL';
  }
  return status === 'ACTIVE';
};
```

- **filteredAndSortedData:** This is a **memoized** value created with useMemo. It efficiently filters the data based on the searchTerm and sorts it according to sortConfig. The component's table will always display this processed data.

```
const filteredAndSortedData = useMemo(() => {
  let processedData = [...data];

  // Apply search filter
  if (searchTerm) {
    const searchLower = searchTerm.toLowerCase();
    processedData = processedData.filter(item => {
      const group = item.Group;
      return (
        group.GroupID.toLowerCase().includes(searchLower) ||
        group.GroupName.toLowerCase().includes(searchLower) ||
        group.GroupRM.toLowerCase().includes(searchLower) ||
        group.GroupRMEmail.toLowerCase().includes(searchLower) ||
        item.Status.toLowerCase().includes(searchLower)
      );
    });
  }

  // Apply sorting
  if (sortConfig) {
    processedData.sort((a, b) => {
      let aValue;
      let bValue;
      if (sortConfig.key === 'Status') {
        aValue = a.Status;
```

```
        bValue = b.Status;
      } else {
        aValue = a.Group[sortConfig.key as keyof Group];
        bValue = b.Group[sortConfig.key as keyof Group];
      }
      if (aValue < bValue) return sortConfig.direction === 'asc' ? -1 : 1;
      if (aValue > bValue) return sortConfig.direction === 'asc' ? 1 : -1;
      return 0;
    });
  }
  return processedData;
}, [data, searchTerm, sortConfig]);
```

6.Rendering the UI

The return statement renders the actual user interface.

- **Conditional Rendering**: It first checks for loading or error states and displays a placeholder message or skeleton UI.

```
if (loading) {
  return (
    <StyledWrapper>
      <div className="skeleton row" />
      <div className="skeleton row" />
      <div className="skeleton row" />
    </StyledWrapper>
  );
}
```

- **Card Structure**: The content is wrapped in Pega's Card components to maintain the platform's look and feel.

```
<Card>
      <CardHeader>Master Data Hierarchy</CardHeader>
      <CardContent>
       Header and Action Container Code
       Hierarchical Table Container Code
       </CardContent>
</Card>
```

- **Header and Actions**: It displays the title and a row of controls, including the **search bar** (Input) and **buttons** for "Bulk Upload" and "Bulk Download". The Bulk Upload button is disabled for the 'reviewmanager' role.

```
<div className="button-group">
          <Button
            variant="primary"
            onClick={() => {
              const caseTypeID = ABC-MasterData-Work-BulkUpload';
              const containerName = 'primary';
```

```
    const actions = getPConnect().getActionsApi();

    actions.createWork(caseTypeID, { containerName });
   }}
   disabled={operatorId === 'reviewmanager'}
  >
   Bulk Upload
  </Button>
  <Button
   variant="secondary"
   onClick={handleBulkDownload}
   disabled={isDownloading}
  >
   {isDownloading ? 'Downloading...' : 'Bulk Download'}
  </Button></div>
```

Hierarchical Table: The main table displays the data.

- The header row allows **sorting** by clicking on column names.
- The body iterates through the filteredAndSortedData. Each row shows the main Group-level information.
- The last two columns are for actions: an **edit icon** that opens a case and an **expand/collapse icon** that toggles the nested tables.

```
  <div className="table-container">
 <table>
  <thead>
   <tr>
    <th onClick={() => handleSort('GroupID')}>
     Group Code {sortConfig?.key === 'GroupID' && (sortConfig.direction === 'asc' ? '↑' : '↓')}
    </th>
    <th onClick={() => handleSort('GroupName')}>
     Group Name {sortConfig?.key === 'GroupName' && (sortConfig.direction === 'asc' ? '↑' : '↓')}
    </th>
    <th onClick={() => handleSort('GroupRM')}>
 Global Account Manager{sortConfig?.key === 'GroupRM' && (sortConfig.direction === 'asc' ? '↑' : '↓')}
    </th>
                <th onClick={() => handleSort('GroupRMEmail')}>
 Global Account Manager Email {sortConfig?.key === 'GroupRMEmail' && (sortConfig.direction ===
'asc' ? '↑' : '↓')}
    </th>
    <th onClick={() => handleSort('CINs')}>
     CINs {sortConfig?.key === 'CINs' && (sortConfig.direction === 'asc' ? '↑' : '↓')}
    </th>
    <th onClick={() => handleSort('GPSClientTier')}>
     GPS Client Tier {sortConfig?.key === 'GPSClientTier' && (sortConfig.direction === 'asc' ? '↑' :
'↓')}
    </th>
    <th onClick={() => handleSort('Status')}>
     Status {sortConfig?.key === 'Status' && (sortConfig.direction === 'asc' ? '↑' : '↓')}
    </th>
    <th>Edit</th>
   </tr>
  </thead>
  <tbody>
   {filteredAndSortedData.map((item) => (
    <React.Fragment key={item.Group.GroupID}>
```

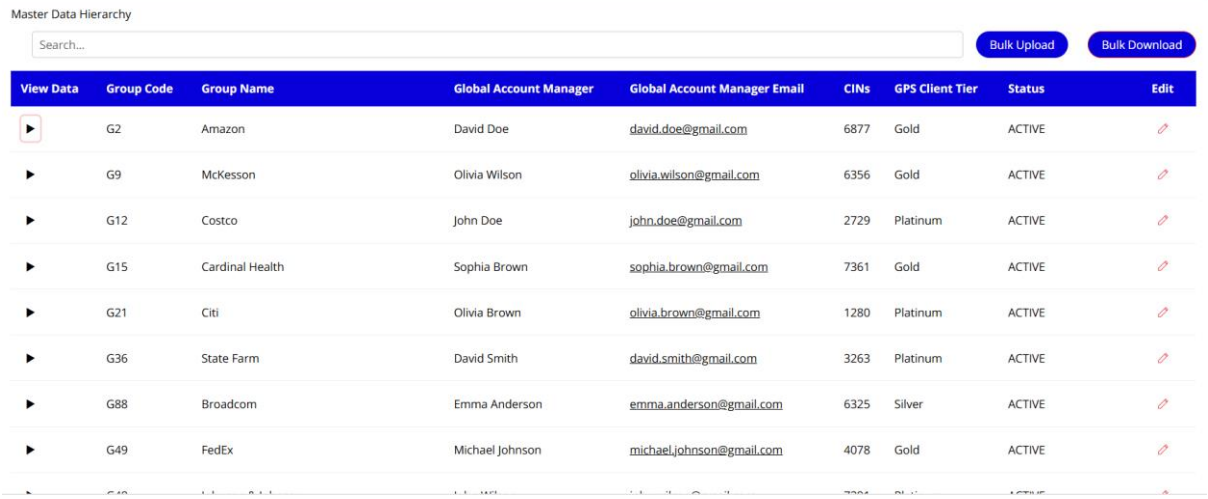
```
<tr>
  <td>{getActiveOrRegularData(item).group.GroupID}</td>
  <td>{getActiveOrRegularData(item).group.GroupName}</td>
  <td>{getActiveOrRegularData(item).group.GroupRM}</td>
  <td>
  <span className="email-link">{getActiveOrRegularData(item).group.GroupRMEmail}</span>
  </td>
  <td>{getActiveOrRegularData(item).group.CINs}</td>
  <td>{getActiveOrRegularData(item).group.GPSClientTier}</td>
  <td>{item.Status}</td>
  <td>
    {shouldShowEditIcon(item.Status) && (
      <button
        type="button"
        className="icon-button"
        onClick={() => handleCaseOpen(item)}
        aria-label="Edit"
      >
        <Icon name="pencil" />
      </button>
    )}
  </td>
  <td>
    <button
      type="button"
      className="icon-button"
      onClick={() => toggleRow(item.Group.GroupID)}
      aria-label="Toggle details"
    >
      {expandedRows.has(item.Group.GroupID) ? '▼' : '▶'}
    </button>
  </td>
</tr>
{expandedRows.has(item.Group.GroupID) && (
  <tr>
    <td colSpan={8}>
      <div className="expanded-content">
        <div className="table-section">
          <h4>Regions</h4>
          <table>
            <thead>
              <tr>
                <th>Region Name</th>
                <th>Region RM</th>
                <th>Region RM Email</th>
              </tr>
            </thead>
            <tbody>
              {getActiveOrRegularData(item).regions.map((region: Region) => (
                <tr key={`${item.Group.GroupID}-${region.RegionName}`}>
                  <td>{region.RegionName}</td>
                  <td>{region.RegionRM}</td>
                  <td>
                    <span className="email-link">{region.RegionRMEmail}</span>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
```

```
                </div>
                <div className="table-section">
                  <h4>Countries</h4>
                  <table>
                    <thead>
                      <tr>
                       <th>Country Name</th>
                        <th>Country RM</th>
                        <th>Country RM Email</th>
                        <th>Service Mailbox</th>
                      </tr>
                    </thead>
                    <tbody>
                      {getActiveOrRegularData(item).countries.map((country: Country) => (
                        <tr key={`${item.Group.GroupID}-${country.CountryName}`}>
                          <td>{country.CountryName}</td>
                          <td>{country.CountryRM}</td>
                          <td>
                           <span className="email- link">{country.CountryRMEmail}</span>
                          </td>
                          <td>
                            <span className="email-link">{country.ServicingEmail}</span>
                          </td>
                        </tr>
                      ))}
                    </tbody>
                  </table>
                </div>
              </div>
            </td>
          </tr>
        )}
      </React.Fragment>
    ))}
  </tbody>
 </table>
</div>
```

- **Nested Tables**: When a row is expanded, it renders a new table row with colSpan=8. Inside this, there are two separate nested tables to display the Regions and Countries associated with the main group, creating the **structured hierarchy**.

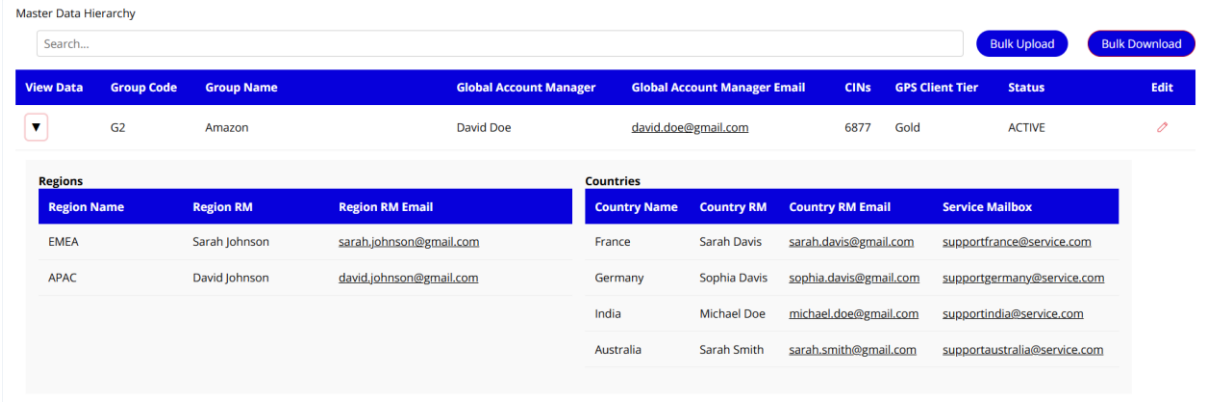
OUTPUT SCREENS:

Master List Table Widget Component



View Data	Group Code	Group Name	Global Account Manager	Global Account Manager Email	CINs	GPS Client Tier	Status	Edit
▶	G2	Amazon	David Doe	david.doe@gmail.com	6877	Gold	ACTIVE	
▶	G9	McKesson	Olivia Wilson	olivia.wilson@gmail.com	6356	Gold	ACTIVE	
▶	G12	Costco	John Doe	john.doe@gmail.com	2729	Platinum	ACTIVE	
▶	G15	Cardinal Health	Sophia Brown	sophia.brown@gmail.com	7361	Gold	ACTIVE	
▶	G21	Citi	Olivia Brown	olivia.brown@gmail.com	1280	Platinum	ACTIVE	
▶	G36	State Farm	David Smith	david.smith@gmail.com	3263	Platinum	ACTIVE	
▶	G88	Broadcom	Emma Anderson	emma.anderson@gmail.com	6325	Silver	ACTIVE	
▶	G49	FedEx	Michael Johnson	michael.johnson@gmail.com	4078	Gold	ACTIVE	

Expanding Hierarchy Master List with Corresponding Regions and Countries of Group



View Data	Group Code	Group Name	Global Account Manager	Global Account Manager Email	CINs	GPS Client Tier	Status	Edit
▼	G2	Amazon	David Doe	david.doe@gmail.com	6877	Gold	ACTIVE	

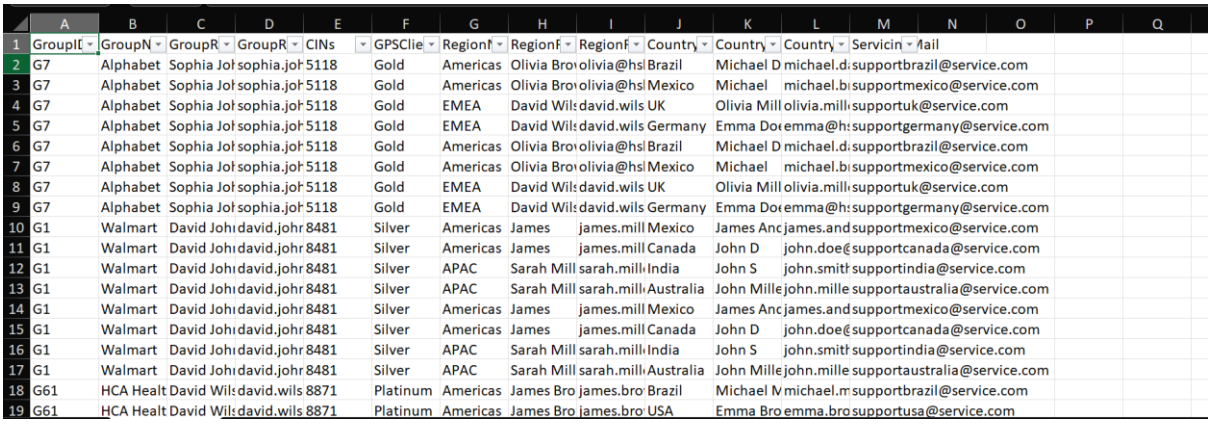
Regions

Region Name	Region RM	Region RM Email
EMEA	Sarah Johnson	sarah.johnson@gmail.com
APAC	David Johnson	david.johnson@gmail.com

Countries

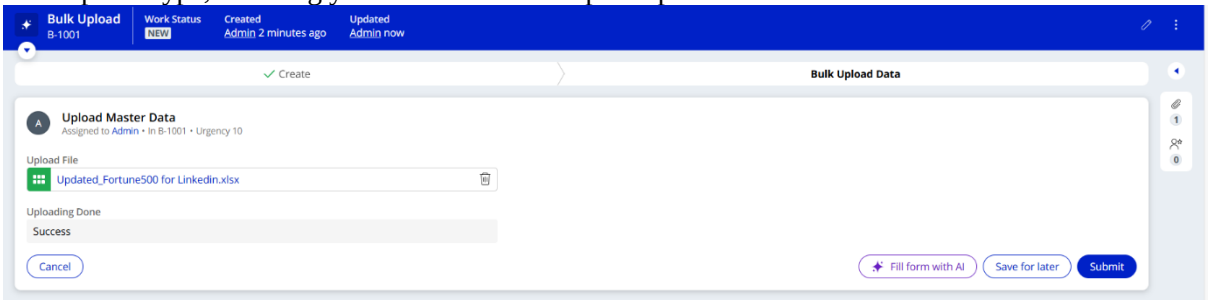
Country Name	Country RM	Country RM Email	Service Mailbox
France	Sarah Davis	sarah.davis@gmail.com	supportfrance@service.com
Germany	Sophia Davis	sophia.davis@gmail.com	supportgermany@service.com
India	Michael Doe	michael.doe@gmail.com	supportindia@service.com
Australia	Sarah Smith	sarah.smith@gmail.com	supportaustralia@service.com

The generated Excel file includes an active filtering feature on all columns by default, allowing users to easily sort and filter the data.

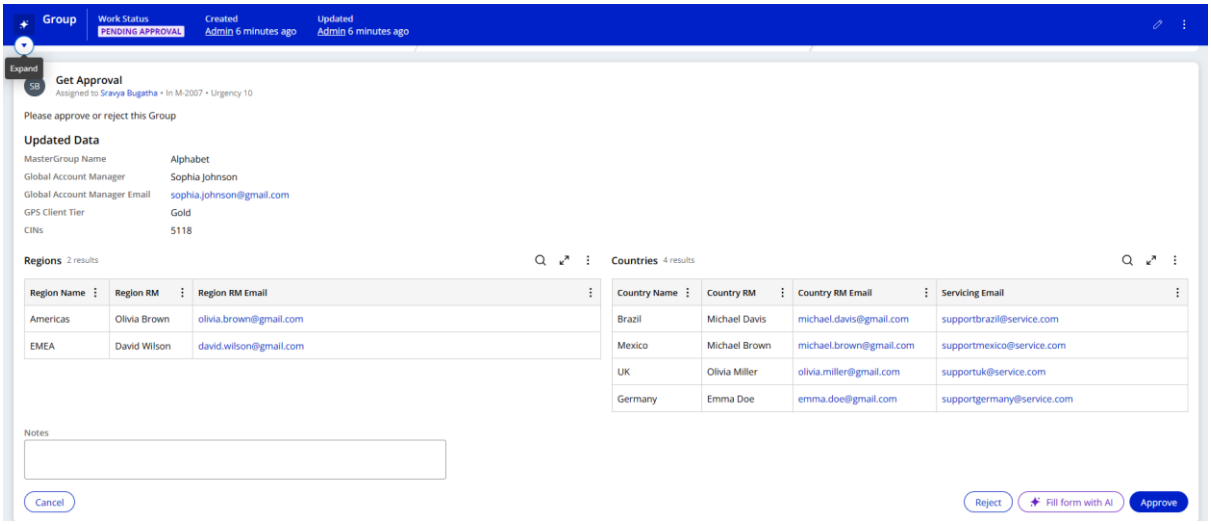


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	GroupID	GroupN	GroupR	GroupR	CINs	GPSClie	RegionF	RegionF	RegionF	Country	Country	Country	Servicin	Mail			
2	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	Americas	Olivia Bro	olivia@hs	Brazil	Michael D	michael.d	supportbrazil@service.com				
3	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	Americas	Olivia Bro	olivia@hs	Mexico	Michael	michael.b	supportmexico@service.com				
4	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	EMEA	David Wils	david.wils	UK	Olivia Mill	olivia.mill	supportuk@service.com				
5	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	EMEA	David Wils	david.wils	Germany	Emma Doe	emma@hs	supportgermany@service.com				
6	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	Americas	Olivia Bro	olivia@hs	Brazil	Michael D	michael.d	supportbrazil@service.com				
7	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	Americas	Olivia Bro	olivia@hs	Mexico	Michael	michael.b	supportmexico@service.com				
8	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	EMEA	David Wils	david.wils	UK	Olivia Mill	olivia.mill	supportuk@service.com				
9	G7	Alphabet	Sophia Jol	sophia.jol	5118	Gold	EMEA	David Wils	david.wils	Germany	Emma Doe	emma@hs	supportgermany@service.com				
10	G1	Walmart	David Johi	david.johr	8481	Silver	Americas	James	james.mill	Mexico	James Anc	james.and	supportmexico@service.com				
11	G1	Walmart	David Johi	david.johr	8481	Silver	Americas	James	james.mill	Canada	John D	john.doe@	supportcanada@service.com				
12	G1	Walmart	David Johi	david.johr	8481	Silver	APAC	Sarah Mill	sarah.mill	India	John S	john.smit	supportindia@service.com				
13	G1	Walmart	David Johi	david.johr	8481	Silver	APAC	Sarah Mill	sarah.mill	Australia	John Mille	john.mille	supportaustralia@service.com				
14	G1	Walmart	David Johi	david.johr	8481	Silver	Americas	James	james.mill	Mexico	James Anc	james.and	supportmexico@service.com				
15	G1	Walmart	David Johi	david.johr	8481	Silver	Americas	James	james.mill	Canada	John D	john.doe@	supportcanada@service.com				
16	G1	Walmart	David Johi	david.johr	8481	Silver	APAC	Sarah Mill	sarah.mill	India	John S	john.smit	supportindia@service.com				
17	G1	Walmart	David Johi	david.johr	8481	Silver	APAC	Sarah Mill	sarah.mill	Australia	John Mille	john.mille	supportaustralia@service.com				
18	G61	HCA Healt	David Wils	david.wils	8871	Platinum	Americas	James Bro	james.bro	Brazil	Michael M	michael.m	supportbrazil@service.com				
19	G61	HCA Healt	David Wils	david.wils	8871	Platinum	Americas	James Bro	james.bro	USA	Emma Bro	emma.bro	supportusa@service.com				

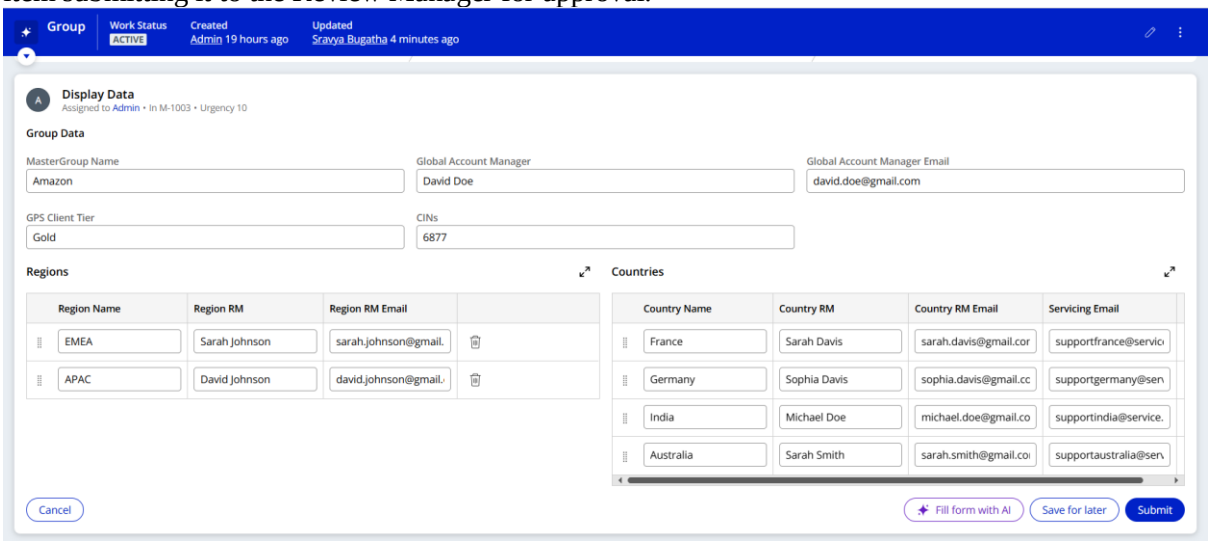
Clicking the **"Bulk Upload"** button immediately opens a new Pega case of the ABC-MasterData-Work-BulkUpload type, allowing you to start the bulk upload process.



When the Review Manager clicks the edit icon, the system directly opens the assignment for them to approve.



When the Admin clicks the edit icon, an assignment opens directly, allowing them to make changes or delete an item submitting it to the Review Manager for approval.



7th floor, Mahaveer The
Watermark, The Water Mark,
Whitefields, HITEC City,
Kondapur, Telangana 500081

email: contact@truviq.com
website: www.truviq.com