Complete code can found here : renderAdmin.xsl & AdminXMLController.php

AdminXMLController

```php
class AdminXMLController extends Controller
{
    public function renderAdmins()
    {
        $admins = Administrator::all()->toArray();

        $array = [
            'admin' => [
                [$admins]
            ],
        ];

        ### Load XML file
        $xml = new \DOMDocument;
        $xml->loadXML(ArrayToXml::convert($array, 'administrators'));

        ### Load XSL file
        $xsl = new \DOMDocument;
        $xsl->load('xsl\renderAdmin.xsl');

        ### Configure the transformer
        $proc = new \XSLTProcessor;

        ### Attach the xsl rules
        $proc->importStyleSheet($xsl);

        ### Transform
        echo $proc->transformToXML($xml);
    }
}
```

Code	Explanation

`$admins = Administrator::all()->toArray();`
 	
`$array = [`
`'admin' => [`
`[$admins]`
`],`
`];`
	Convert the Eloquent collection of administrator to an array

`$xml = new \DOMDocument;`	Create a new DOM Document to hold the web page structure
`$xml->loadXML(ArrayToXml::convert($array, 'administrators'));`	Load the XML data and set the custom root name to administrators Notes: we have to change the array to XML

	format [1]. In this case, spatial/array-to-xml is installed to convert the array to xml (A simple class to convert an array to xml)
`$xsl = new \DOMDocument;`	Create a new DOM Document to hold our web page structure.
`$xsl->load('xsl\renderAdmin.xsl');`	Load the XSL file
`$proc = new \XSLTProcessor;`	Configure the XSLT Processor to transform the XSLT stylesheet to an XML document which will produce a new XML document
`$proc->importStyleSheet($xsl);`	Import the stylesheet into the XSLTProcessor for transformations
`echo $proc->transformToXML($xml);`	Produce a valid XHTML output and display it out

```
<?xml version="1.0"?>\n
<administrators>\n
  <admin>\n
    <id>1</id>\n
    <user_id>A1000</user_id>\n
    <email>zixuan2001711@gmail.com</email>\n
    <password>$2y$10$VqjWVOI9PgSqNjGb1WsEq.OY10JGm7oHBs99L65xOqbHH9p9KYCDG</password>\n
    <user_name>Loo Zi Xuan</user_name>\n
    <contact_no>01108981221</contact_no>\n
    <address>87, Taman Ligamas, 44300 Selangor</address>\n
    <admin_role>super_admin</admin_role>\n
    <profile_image>name.png</profile_image>\n
    <status>Active</status>\n
    <created_at/>\n
    <updated_at/>\n
  </admin>\n
  <admin>\n
    <id>2</id>\n
    <user_id>A1001</user_id>\n
    <email>yuenkuan@gmail.com</email>\n
    <password>$2y$10$WoB7jZ8mQWssBcQphfT6SOMau5wFOS4GLCX3hAp4HsCaRKMRED6o.</password>\n
    <user_name>Yuen Kuan</user_name>\n
    <contact_no>0123456789</contact_no>\n
    <address>23 kampung gurney ulu yam bharu</address>\n
    <admin_role>normal_admin</admin_role>\n
    <profile_image>Kampung.jpeg</profile_image>\n
    <status>Active</status>\n
    <created_at>2022-03-19T23:10:56.000000Z</created_at>\n
    <updated_at>2022-03-25T18:13:42.000000Z</updated_at>\n
  </admin>\n
  <admin>\n
    <id>3</id>\n
    <user_id>A1002</user_id>\n
    <email>jialok@gmail.com</email>\n
    <password>$2y$10$HtYjsq8yUnGBrLB6/Hf0GeeWCMUvm1.KauSIIXjV2NOkmMolalEKS</password>\n
    <user_name>kah lok</user_name>\n
    <contact_no>01118981229</contact_no>\n
    <address>53 kampung gurney ulu yam bharu</address>\n
    <admin_role>super_admin</admin_role>\n
    <profile_image>ai.png</profile_image>\n
    <status>Active</status>\n
    <created_at>2022-03-20T16:31:20.000000Z</created_at>\n
    <updated_at>2022-03-21T20:01:55.000000Z</updated_at>\n
  </admin>\n
</administrators>\n
```


[1] XML view from dd() helper function

Afterward, I define a **renderAdmin.xsl** that includes XPath expressions to define the XML document transformation and presentation.

Output

First Tab: Super Admin

Administrator Table Information

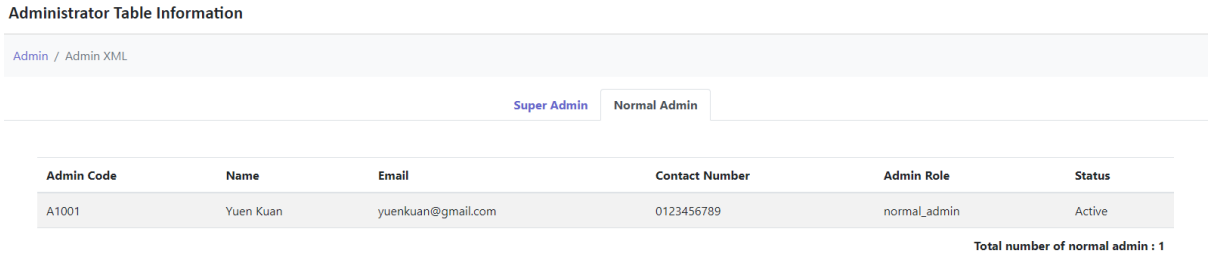
Admin / Admin XML

Super Admin | Normal Admin

No.	Admin Code	Name	Email	Contact Number	Admin Role	Status
1.	A1000	Loo Zi Xuan	zixuan2001711@gmail.com	01108981221	super_admin	Active
3.	A1002	kah lok	jialok@gmail.com	01118981229	super_admin	Active

Total number of super admin : 2

Second Tab: Normal Admin



XPath expression used

Path Expression	Result
First Tab: Super Admin	
`administrators/admin[admin_role='super_admin']`	Select all the admin element of the administrators element that have a admin role element with value ‘super_admin’
`<xsl:if test="status = 'Active'">`	A conditional test against the content of XML and display the data if the value of status element is ‘Active’
`count(//admin[admin_role='super_admin' and status='Active'])`	Return the total numbers of admin element that have the admin role element with value ‘super_admin’ and a status element with value ‘Active’
Second Tab: Normal Admin	
`administrators/admin[admin_role='normal_admin']`	Select all the admin element of the administrators element that have a admin role element with value ‘normal_admin’
`<xsl:if test="status = 'Active'">`	A conditional test against the content of XML and display the data if the value of status element is ‘Active’
`count(//admin[admin_role='normal_admin' and status='Active'])`	Return the total numbers of admin element that have the admin role element with value ‘normal_admin’ and a status element with value ‘Active’

XSLT StyleSheet

- **<xsl:for-each>** to process a number of items under path `administrators/admin` and loop them out in a tabular format.
- **<xsl:if>** to get the XML data if the value of status element is ‘Active’.
- **<xsl:value-of>** to evaluate the expression and display it

- **<xsl:number>** to number the part of XML document

```
<xsl:for-each select="administrators/admin[admin_role='super_admin']">
    <xsl:if test="status = 'Active'">
        <tr>
            <td>
                <xsl:number format="1. "/>
            </td>
            <td>
                <xsl:value-of select="user_id" />
            </td>
            <td>
                <xsl:value-of select="user_name" />
            </td>
            <td>
                <xsl:value-of select="email" />
            </td>
            <td>
                <xsl:value-of select="contact_no" />
            </td>
            <td>
                <xsl:value-of select="admin_role" />
            </td>
            <td>
                <xsl:value-of select="status" />
            </td>
        </tr>
    </xsl:if>
</xsl:for-each>
```

```
<td class="text-right" colspan="7">
    <b>Total number of super admin :
        <xsl:value-of select="count(//admin[admin_role='super_admin' and status='Active'])"/>
    </b>
</td>
```

First tab: Super Admin

Note: The implementation of second tab (normal admin) is similar with first tab (super admin)