

Technical Documentation

Document ID:5459 and 5457

Document Version:1.0

Authors:Shashank G Suvarna and Pooja Bhat K

Team-7

1. Overall Description

This document contains all the functionalities that we successfully finished. Here we are creating the Career Page with all working functionalities like the admin can add the available jobs in backend, if the vacancies are closed the admin can disable the available job in the backend this is achieved using grid and form. In frontend, the user or job applicant can view all the available jobs and view specific job requirement and responsibility and apply for that job. When the applicant applies for the job the email must be generated from applicant to the admin. This email sending is achieved through SMTP.

2. Technical Requirements

2.1 GRID and FORM

Step 1: Creating the Module: In magneto root directory app/code, create the folder to build the extension in a given structure which is defined as in this project the vendor name as ‘News’ and module name as ‘Career’.

app/code/News/Career/etc/module.xml

```
News > Career > etc > module.xml
1  <?xml version="1.0"?>
2  <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd">
3  <module name="News_Career" >
4      <sequence>
5          <module name="Magento_Directory"/>
6          <module name="Magento_Config"/>
7      </sequence>
8  </module>
9 </config>
10
```

Step 2: Create the acl.xml: Magento 2 admin acl use an authentication system and a robust system for create Access Control List Rules (ACL) .

app/code/News/Career/etc/acl.xml

```

News > Career > etc > acl.xml
1  <?xml version="1.0"?>
2  <!--
3  /**
4  * Copyright © Magento, Inc. All rights reserved.
5  * See COPYING.txt for license details.
6  */
7 -->
8 <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="urn:magento:framework:Acl/etc/acl.xsd">
9   <acl>
10    <resources>
11      <resource id="Magento_Backend::admin">
12          <resource id="News_Career::career" title="Career" sortOrder="0">
13              <resource id="News_Career::manage" title="All Careers" sortOrder="10">
14                  </resource>
15          </resource>
16      </resource>
17  </resources>
18 </acl>
19 </config>
21

```

Step 3: Create the db_schema.xml: This file is used to create the new tables or maintain the tables

app/code/News/Career/etc/db_schema.xml

```

News > Career > etc > db_schema.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2
3 <schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="urn:magento:framework:Setup/Declaration/Schema/etc/schema.xsd">
4   <table name="career_data" resource="default" engine="innodb" comment="career_data">
5
6     <column xsi:type="smallint" name="id" padding="7" unsigned="false" nullable="false" identity="true" comment="ID" />
7     <column xsi:type="varchar" name="job_role" nullable="false" length="20" comment="job_role" />
8     <column xsi:type="varchar" name="place" nullable="false" length="20" comment="Place" />
9     <column xsi:type="varchar" name="email" nullable="false" length="50" comment="Email" />
10    <column xsi:type="int" name="experience" nullable="false" length="20" comment="experience" />
11    <column xsi:type="text" name="job_description" nullable="false" length="20" comment="Job description" />
12    <column xsi:type="text" name="job_responsibility" nullable="false" length="20" comment="Job responsibility" />
13    <column xsi:type="text" name="available" nullable="false" length="20" comment="available" />
14
15    <constraint xsi:type="primary" referenceId="PRIMARY">
16      <column name="id" />
17    </constraint>
18  </table>
19 </schema>

```

Step 4: Create the di.xml: This file configures which dependencies are injected by object manager and the application reads all the di.xml configuration files which will be merged all together by appending all nodes.

app/code/News/Career/etc/di.xml

```

News > Career > etc > di.xml
1  <?xml version="1.0"?>
2  <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">
3  <type name="Magento\Framework\View\Element\UiComponent\DataProvider\CollectionFactory">
4      <arguments>
5          <argument name="collections" xsi:type="array">
6              <item name="career_listing_data_source"
xsi:type="string">News\Career\Model\ResourceModel\Sample\Grid\Collection</item>
7          </argument>
8      </arguments>
9  </type>
10 <type name="News\Career\Model\ResourceModel\Sample\Grid\Collection">
11     <arguments>
12         <argument name="mainTable" xsi:type="string">career_data</argument>
13         <argument name="eventPrefix" xsi:type="string">sample_grid_collection</argument>
14         <argument name="eventObject" xsi:type="string">sample_grid_collection</argument>
15         <argument name="resourceModel" xsi:type="string">News\Career\Model\ResourceModel\Sample</argument>
16     </arguments>
17 </type>
18
19
20 </config>
21

```

Step 5: Create the routes.xml in admin html: The route will define the name for the module which will be used in the url to find the module and execute the controller action.

app/code/News/Career/etc/adminhtml/routes.xml

```

News > Career > etc > adminhtml > routes.xml
1  <?xml version="1.0" ?>
2  <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="urn:magento:framework\App/etc/routes.xsd">
3      <router id="admin">
4          <route frontName="career" id="career">
5              <module name="News_Career" before="Magento_Backend"/>
6          </route>
7      </router>
8  </config>
9

```

Step 6: Create the registration.php: It is used to notify to magento application about the module in the system. It registers the module.

app/code/News/Career/registration.php

News > Career > 🏠 registration.php

```

1  <?php
2  \Magento\Framework\Component\ComponentRegistrar::register(
3      \Magento\Framework\Component\ComponentRegistrar::MODULE,
4      'News_Career',
5      __DIR__
6  );
7

```

Step 7: Create the menu.xml : It is located in module's etc/adminhtml folder which consists of menu nodes, config and add multiple directives.

app/code/News/Career/etc/adminhtml/menu.xml

```

News > Career > etc > adminhtml > 📄 menu.xml
1  <?xml version="1.0" ?>
2  <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:App/etc/routes.xsd">
3  <menu>
4  <add id="News_Career::career"
5  title="Career" translate="title"
6  module="News_Career"
7  sortOrder="20"
8  resource="News_Career::career"/>
9  <add id="News_Career::manage" title="All Careers" translate="title" module="News_Career" sortOrder="10"
parent="News_Career::career" action="career/sample" resource="News_Career::manage"/>
10
11 </menu>
12 </config>
13

```

Step 8: Create the Sample.php in Model: The Sample.php file in the model folder is a PHP class that represents the data model for a specific entity in the system. This file is typically used to define the business logic that relates to that entity, such as saving and retrieving data from the database.

app/code/News/Career/Model/Sample.php

News > Career > Model > Sample.php

```
1 <?php
2 namespace News\Career\Model;
3 use Magento\Framework\Model\AbstractModel;
4
5 class Sample extends AbstractModel
6 {
7     protected function _construct()
8     {
9         $this->_init('News\Career\Model\ResourceModel\Sample');
10    }
11 }
12
```

Step 9: Create the Sample.php in Resource Model folder:

app/code/News/Career/Model/ResourceModel/Sample.php

```
News > Career > Model > ResourceModel > Sample.php
1 <?php
2 namespace News\Career\Model\ResourceModel;
3
4
5 class Sample extends \Magento\Framework\Model\ResourceModel\Db\AbstractDb{
6     protected function _construct()
7     {
8         $this->_init('career_data','id');
9     }
10}
11
```

Step 10: Create the Collection.php in Model-Resource Model-Sample folder: The Collection class in Magento 2 is used to retrieve a set of data from the database. The Collection class is typically used to work with data from custom database tables or to extend core Magento 2 models and work with data from their associated database tables.

app/code/News/Career/Model/ResourceModel/Sample/Collection.php

News > Career > Model > ResourceModel > Sample > Collection.php

```
1  <?php
2  namespace News\Career\Model\ResourceModel\Sample;
3  use Magento\Framework\Model\ResourceModel\Db\Collection\AbstractCollection;
4
5
6
7  class Collection extends AbstractCollection
8  {
9      /**
10      * @var string
11      */
12      protected $id='id';
13
14      protected function _construct()
15      {
16          $this->_init(
17              'News\Career\Model\Sample',
18              'News\Career\Model\ResourceModel\Sample'
19          );
20
21      }
22  }
```

Step 11: Create the Collection.php in Model-ResourceModel-Sample-Grid folder: In Magento 2, the Collection class for a grid is used to retrieve a set of data that will be displayed in the grid. The Collection class is typically defined in a file called Collection.php, which is located in the Model/ResourceModel directory of a module.

app/code/News/Career/Model/ResourceModel/Sample/Grid/Collection.php

```

News > Career > Model > ResourceModel > Sample > Grid > Collection.php
1  <?php
2  /**
3   * Copyright © Magento, Inc. All rights reserved.
4   * See COPYING.txt for license details.
5   */
6  namespace News\Career\Model\ResourceModel\Sample\Grid;
7
8  use Magento\Framework\Api\Search\SearchResultInterface;
9  use Magento\Framework\Api\Search\AggregationInterface;
10 use News\Career\Model\ResourceModel\Sample\Collection as BlockCollection;
11 use Magento\Framework\App\ObjectManager;
12 use Magento\Framework\Data\Collection\Db\FetchStrategyInterface;
13 use Magento\Framework\Data\Collection\EntityFactoryInterface;
14 use Magento\Framework\DB\Adapter\AdapterInterface;
15 use Magento\Framework\Event\ManagerInterface;
16 use Magento\Framework\Model\ResourceModel\Db\AbstractDb;
17 use Magento\Framework\Stdlib\DateTime\TimezoneInterface;
18 use Magento\Store\Model\StoreManagerInterface;
19 use Psr\Log\LoggerInterface;
20
21 /**
22  * Collection for displaying grid of career| blocks
23 */
24 class Collection extends BlockCollection implements SearchResultInterface
25 {
    ...
}

```

```

News > Career > Model > ResourceModel > Sample > Grid > Collection.php
35
36 /**
37  * @param EntityFactoryInterface $entityFactory
38  * @param LoggerInterface $logger
39  * @param FetchStrategyInterface $fetchStrategy
40  * @param ManagerInterface $eventManager
41  * @param string $mainTable
42  * @param string $eventPrefix
43  * @param string $eventObject
44  * @param string $resourceModel
45  * @param string $model
46  * @param AdapterInterface|string|null $connection
47  * @param AbstractDb $resource
48  * @param TimezoneInterface|null $timeZone
49
50  * @SuppressWarnings(PHPMD.ExcessiveParameterList)
51 */
52 public function __construct(
53     EntityFactoryInterface $entityFactory,
54     LoggerInterface $logger,
55     FetchStrategyInterface $fetchStrategy,
56     ManagerInterface $eventManager,
57     $mainTable,
58     $eventPrefix,
59     $eventObject,
60     $resourceModel,
61     $model = \Magento\Framework\View\Element\UiComponent\DataProvider\Document::class,
62     $connection = null,
63     AbstractDb $resource = null,
64     TimezoneInterface $timeZone = null
65 )
66 {
67     parent::__construct(
68         $entityFactory,
69         $logger,
70         $fetchStrategy,
71         $eventManager,
72         $connection,
73         $resource
74     );
75     $this->_eventPrefix = $eventPrefix;
76     $this->_eventObject = $eventObject;
}

```

```

News > Career > Model > ResourceModel > Sample > Grid > Collection.php
75     $this->_eventObject = $eventObject;
76     $this->_init($model, $resourceModel);
77     $this->setMainTable($mainTable);
78     $this->timeZone = $timeZone ?: ObjectManager::getInstance()->get(TimezoneInterface::class);
79 }
80 /**
81 * @inheritDoc
82 */
83 public function addFieldToFilter($field, $condition = null)
84 {
85     if ($field === 'creation_time' || $field === 'update_time') {
86         if (is_array($condition)) {
87             foreach ($condition as $key => $value) {
88                 $condition[$key] = $this->timeZone->convertConfigTimeToUtc($value);
89             }
90         }
91     }
92     return parent::addFieldToFilter($field, $condition);
93 }
94 /**
95 * Get aggregation interface instance
96 *
97 * @return AggregationInterface
98 */
99 public function getAggregations()
100 {
101     return $this->aggregations;
102 }
103 /**
104 * Set aggregation interface instance
105 *
106 * @param AggregationInterface $aggregations
107 * @return $this
108 */
109 public function setAggregations($aggregations)
110 {
111     $this->aggregations = $aggregations;
112     return $this;
113 }
114 /**
115 * Get search criteria.
116 *
117 * @return \Magento\Framework\Api\SearchCriteriaInterface|null
118 */
119 public function getSearchCriteria()
120 {
121     return null;
122 }
123 /**
124 * Set search criteria.
125 *
126 * @param \Magento\Framework\Api\SearchCriteriaInterface $searchCriteria
127 * @return $this
128 * @SuppressWarnings(PHPMD.UnusedFormalParameter)
129 */
130 public function setSearchCriteria(\Magento\Framework\Api\SearchCriteriaInterface $searchCriteria = null)
131 {
132     return $this;
133 }
134 /**
135 * Get total count.
136 *
137 * @return int
138 */
139 public function getTotalCount()
140 {
141     return $this->getSize();
142 }
143 /**
144 * Set total count.
145 *
146 * @param int $totalCount
147 * @return $this
148 * @SuppressWarnings(PHPMD.UnusedFormalParameter)
149 */
150 public function setTotalCount($totalCount)
151 {
152     return $this;
153 }
154 /**
155 * Set items list.
156 *
157 * @param \Magento\Framework\Api\ExtensibleDataInterface[] $items
158 * @return $this
159 * @SuppressWarnings(PHPMD.UnusedFormalParameter)
160 */
161 public function setItems(array $items = null)
162 {
163     return $this;
164 }
165 }
166 
```

Step 12: Create the DataProvider.php: The DataProvider class for a grid is responsible for retrieving data from a collection object and preparing it for display in the grid. The DataProvider class is typically defined in a file called DataProvider.php, which is located in the model directory of a module.

app/code/News/Career/Model/Grid/DataProvider.php

```

News > Career > Model > Grid > DataProvider.php
1  <?php
2
3  namespace News\Career\Model\Grid;
4
5  use News\Career\Model\ResourceModel\Sample\CollectionFactory;
6  use Magento\Ui\DataProvider\AbstractDataProvider;
7
8  class DataProvider extends AbstractDataProvider
9  {
10     protected $loadedData;
11
12     public function __construct(
13         $name,
14         $primaryFieldName,
15         $requestFieldName,
16         CollectionFactory $collectionFactory,
17         array $meta = [],
18         array $data = []
19     ) {
20         $this->collection = $collectionFactory->create();
21         parent::__construct($name, $primaryFieldName, $requestFieldName, $meta, $data);
22     }
23
24     public function getData()
25     {
26         $items = $this->collection->getItems();
27         foreach ($items as $model) {
28             $this->loadedData[$model->getId()] = $model->getData();
29         }
30         return $this->loadedData;
31     }
32 }
33

```

Step 13: Create the index.php in the Controller-Adminhtml:

The index.php file located in the adminhtml directory is the main entry point for the Magento admin panel. It is responsible for initializing the admin application and routing incoming requests to the appropriate controller.

app/code/News/Career/Controller/Adminhtml/Sample/Index.php

```

News > Career > Controller > Adminhtml > Sample > index.php
1  <?php
2
3  namespace News\Career\Controller\Adminhtml\Sample;
4
5  use \Magento\Backend\App\Action;
6  use \Magento\Backend\App\Action\Context;
7  use \Magento\Framework\Registry;
8  use \Magento\Framework\View\Result\PageFactory;
9  use \Magento\Backend\Model\View\Result\ForwardFactory;
10
11 class Index extends Action
12 {
13     /**
14      * Authorization level of a basic admin session
15      *
16      * @see _isAllowed()
17      */
18     const ACTION_RESOURCE='News_Career::career';
19
20     /**
21      * Core registry
22      *
23      * @var Registry
24      */
25     protected $coreRegistry;
26
27     /**
28      * Result PageFactory
29      *
30      * @var PageFactory
31      */
32     protected $resultPageFactory;
33
34     /**
35      * @var ForwardFactory
36      */
37     protected $resultForwardFactory;
38
39     /**
40      * @param Registry $registry
41      * @param PageFactory $resultPageFactory
42      * @param ForwardFactory $resultForwardFactory
43      * @param Context $context
44      */
45     public function __construct(
46         Registry $registry,
47         PageFactory $resultPageFactory,
48         ForwardFactory $resultForwardFactory,
49         Context $context)
50     {
51         $this->coreRegistry = $registry;
52         $this->resultPageFactory = $resultPageFactory;
53
54         $this->resultForwardFactory = $resultForwardFactory;
55         parent::__construct($context);
56     }
57
58     /**
59      * @return \Magento\Framework\View\Result\Page
60      */
61 }

```

Step 14: Create the Save.php file in Controller – Adminhtml:

The save.php file in a custom controller could potentially be responsible for processing a form submission
app/code/News/Career/Controller/Adminhtml/Sample/Save.php

```
News > Career > Controller > Adminhtml > Sample > MassDelete.php
1 <?php
2
3 namespace News\Career\Controller\Adminhtml\Sample;
4
5 use Magento\Backend\App\Action;
6 use Magento\Backend\App\Action\Context;
7 use Magento\Framework\Controller\ResultFactory;
8 use Magento\Ui\Component\MassAction\Filter;
9 use News\Career\Model\ResourceModel\Sample\CollectionFactory;
10
11 class MassDelete extends Action
12 {
13     /**
14      * Mass Action Filter
15      *
16      * @var Filter
17     */
18     protected $filter;
19
20     /**
21      * Collection Factory
22      *
23      * @var CollectionFactory
24     */
25     protected $collectionFactory;
26
27     /**
28      * Constructor
29      *
30      * @param Context $context
31      * @param Filter $filter
32      * @param CollectionFactory $collectionFactory
33     */
34     public function __construct(
35         Context $context,
36         Filter $filter,
37         CollectionFactory $collectionFactory
38     ) {
39         parent::__construct($context);
40         $this->filter = $filter;
41         $this->collectionFactory = $collectionFactory;
42     }
43
44     /**
45      * Execute action
46      *
47      * @return \Magento\Backend\Model\View\Result\Redirect
48     */
49     public function execute()
50     {
51         // Get collection of samples to be deleted
52         $collection = $this->filter->getCollection($this->collectionFactory->create());
53         // echo($collection->getSize());die;
54         // Delete each sample in the collection
55         $deleteCount = 0;
56         foreach ($collection->getItems() as $sample) {
57             $sample->delete();
58             $deleteCount++;
59         }
60
61         // Add success message
62         $this->messageManager->addSuccess(__('A total of %1 record(s) have been deleted.', $deleteCount));
63
64         // Redirect to sample grid
65         $resultRedirect = $this->resultFactory->create(ResultFactory::TYPE_REDIRECT);
66         $resultRedirect->setPath('*/*/index');
67         return $resultRedirect;
68     }
69
70     /**
71      * Check if the current user is allowed to access this action
72      *
73      * @return bool
74     */
75     protected function _isAllowed()
76     {
77         return $this->_authorization->isAllowed('News_Career::sample_delete');
78     }
79
80 }
```

Step 15: Create the Form.php file in Controller folder:

A Form.php file in a custom controller could potentially be responsible for rendering a form for creating or updating and processing a form submission, saving data to the database

app/code/News/Career/Controller/Adminhtml/Sample/Form.php

```

News > Career > Controller > Adminhtml > Sample > Form.php
1  <?php
2
3  namespace News\Career\Controller\Adminhtml\Sample;
4
5  use \Magento\Backend\App\Action;
6  use \Magento\Backend\App\Action\Context;
7  use \Magento\Framework\Registry;
8  use \Magento\Framework\View\Result\PageFactory;
9  use \Magento\Backend\Model\View\Result\ForwardFactory;
10
11 class Form extends Action
12 {
13     /**
14      * Authorization level of a basic admin session
15      *
16      * @see _isAllowed()
17      */
18     const ACTION_RESOURCE='Embitel_Mymodule2::mymodule2';
19
20     /**
21      * Core registry
22      *
23      * @var Registry
24      */
25     protected $coreRegistry;
26
27     /**
28      * Result PageFactory
29      *
30      * @var PageFactory
31      */
32     protected $resultPageFactory;
33
34     /**
35      * @var ForwardFactory
36      */
37     protected $resultForwardFactory;
38
39     /**
40      * @param Registry $registry
41      * @param PageFactory $resultPageFactory
42      * @param ForwardFactory $resultForwardFactory
43      * @param Context $context
44      */
45     public function __construct(
46         Registry $registry,
47         PageFactory $resultPageFactory,
48         ForwardFactory $resultForwardFactory,
49         Context $context)
50     {
51         $this->coreRegistry = $registry;
52         $this->resultPageFactory = $resultPageFactory;
53         $this->resultForwardFactory = $resultForwardFactory;
54         parent::__construct($context);
55     }
56
57 }

```

Step 16: Create the career_listing.xml:

career_listing.xml file is a UI Component file that is used to define the structure and behavior of a listing page for a specific entity type. This file is typically located in the view/adminhtml/ui component directory of a custom module or extension and is used to configure the appearance and functionality of the listing page in the Magento Admin Panel.

app/code/News/Career/view/adminhtml/ui_component/career_listing.xml

```

News > Career > view > adminhtml > ui_component > career_listing.xml
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <!--
 3  /**
 4  * Copyright © Magento, Inc. All rights reserved.
 5  * See COPYING.txt for license details.
 6  */
 7  -->
 8  <listing xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Ui/etc/ui_configuration.xsd">
 9      <argument name="data" xsi:type="array">
10          <item name="js_config" xsi:type="array">
11              <item name="provider" xsi:type="string">career_listing.career_listing_data_source</item>
12          </item>
13      </argument>
14      <settings>
15          <buttons>
16              <button name="add">
17                  <url path="/#/form"/>
18                  <class>primary</class>
19                  <label translate="true">Add New Employee</label>
20              </button>
21          </buttons>
22          <spinner>cms_block_columns</spinner>
23          <deps>
24              <dep>career_listing.career_listing_data_source</dep>
25          </deps>
26      </settings>
27      <dataSource name="career_listing_data_source" component="Magento_Ui/js/grid/provider">
28          <settings>
29              <storageConfig>
30                  <param name="indexField" xsi:type="string">id</param>
31              </storageConfig>
32              <updateUrl path="mui/index/render"/>
33          </settings>
34          <aclResource>News_Career::career</aclResource>
35          <dataProvider class="Magento\Cms\Ui\Component\DataProvider" name="career_listing_data_source">
36              <settings>
37                  <requestFieldName>id</requestFieldName>
38                  <primaryFieldName>id</primaryFieldName>
39              </settings>
40          </dataProvider>
41      </dataSource>
42      <listingToolbar name="listing_top">
43          <settings>
44              <sticky>true</sticky>
45          </settings>
46          <bookmark name="bookmarks"/>
47          <columnsControls name="columns_controls"/>
48          <filterSearch name="fulltext"/>
49          <filters name="listing_filters">
50              <settings>
51                  <templates>
52                      <filters>
53                          <select>
54                              <param name="template" xsi:type="string">ui/grid/filters/elements/ui-select</param>
55                              <param name="component" xsi:type="string">Magento_Ui/js/form/element/ui-select</param>
56                          </select>
57                      </filters>
58                  </templates>
59                  <icon></icon>
60                  <url path="/#/massDelete"></url>
61                  <type>delete</type>
62                  <label translate="true">Delete</label>
63              </settings>
64          </action>
65      </listingToolbar>
66      <paging name="listing_paging"/>
67  </listing>
68  <columns name="cms_block_columns">
69      <selectionsColumn name="ids">
70          <settings>
71              <indexField>id</indexField>
72          </settings>
73      </selectionsColumn>
74      <settings>
75          <editorConfig>
76              <param name="clientConfig" xsi:type="array">
77                  <item name="saveUrl" xsi:type="url" path="/inlineEdit"/>
78                  <item name="validateBeforeSave" xsi:type="boolean">&lt;false&gt;</item>
79              </param>
80              <param name="indexField" xsi:type="string">id</param>
81              <param name="enabled" xsi:type="boolean">true</param>
82              <param name="selectProvider" xsi:type="string">career_listing.career_listing.cms_block_columns.ids</param>
83          </editorConfig>
84          <childDefaults>
85              <param name="fieldAction" xsi:type="array">
86                  <item name="provider" xsi:type="string">career_listing.career_listing.cms_block_columns_editor</item>
87                  <item name="target" xsi:type="string">startEdit</item>
88                  <item name="params" xsi:type="array">
89                      <item name="0" xsi:type="string"><${ $.data.rowIndex }></item>
90                      <item name="1" xsi:type="boolean">true</item>
91                  </array>
92              </param>
93          </childDefaults>
94      </settings>
95      <selectionsColumn name="ids">
96          <settings>
97              <indexField>id</indexField>
98          </settings>
99      </selectionsColumn>
100     <column name="id">
101         <settings>
102             <filter>textRange</filter>
103             <label translate="true">ID</label>
104             <sorting>asc</sorting>
105         </settings>
106     </column>
107     <column name="job_role">
108         <settings>
109             <filter>text</filter>
110             <editor>
111                 <validation>
112                     <rule name="required-entry" xsi:type="boolean">true</rule>
113                 </validation>
114                 <editorType>text</editorType>
115             </editor>
116             <label translate="true">Job Role</label>
117         </settings>
118     </column>
119     <column name="place">
120         <settings>
121             <filter>text</filter>
122             <editor>
123                 <validation>
124                     <rule name="required-entry" xsi:type="boolean">true</rule>
125                     <rule name="no-marginal whitespace" xsi:type="boolean">true</rule>
126                 </validation>
127                 <editorType>text</editorType>
128             </editor>
129             <label translate="true">Place</label>
130         </settings>
131     </column>

```

Step 17: Create the career_form.xml:

The form.xml file defines the structure of the form page by specifying the fields and input elements that should be displayed to the user. It also defines the data sources that should be used to populate the form fields, such as database tables or API endpoints.

app/code/News/

```
49
50     <field name="enable">
51         <argument name="data" xsi:type="array">
52             <item name="options" xsi:type="object">News\Career\Model\Page\Source\EnablingPin</item>
53             <item name="label" xsi:type="string" translate="true">available</item>
54             <item name="label" xsi:type="string" translate="true">available</item>
55             <item name="formElement" xsi:type="string">select</item>
56             <item name="dataScope" xsi:type="string">available</item>
57         </item>
58     </argument><settings>
59     <validation>
60         <rule name="required-entry" xsi:type="boolean">true</rule>
61     </validation>
62     <validation>
63         <rule name="required-entry" xsi:type="boolean">true</rule>
64     </validation>
65   </settings>
66 </field>
67
68     <field name="job_role">
69         <argument name="data" xsi:type="array">
70             <item name="config" xsi:type="array">
71                 <item name="datatype" xsi:type="string">text</item>
72                 <item name="label" xsi:type="string" translate="true">Job Role</item>
73                 <item name="formElement" xsi:type="string">input</item>
74                 <item name="source" xsi:type="string">job_role</item>
75                 <item name="dataScope" xsi:type="string">job_role</item>
76                 <item name="validation" xsi:type="array">
77                     <item name="required-entry" xsi:type="boolean">true</item>
78                 </item>
79             </item>
80         </argument>
81     </field>
82     <field name="place">
83         <argument name="data" xsi:type="array">
84             <item name="config" xsi:type="array">
85                 <item name="datatype" xsi:type="string">text</item>
86                 <item name="label" xsi:type="string" translate="true">Place</item>
87                 <item name="formElement" xsi:type="string">input</item>
88                 <item name="source" xsi:type="string">place</item>
89                 <item name="dataScope" xsi:type="string">place</item>
90                 <item name="validation" xsi:type="array">
91                     <item name="required-entry" xsi:type="boolean">true</item>
92                 </item>
93             </item>
94         </argument>
95     </field>
96     <field name="email">
97         <argument name="data" xsi:type="array">
98             <item name="config" xsi:type="array">
99                 <item name="datatype" xsi:type="string">text</item>
100                 <item name="label" xsi:type="string" translate="true">Email</item>
101                 <item name="formElement" xsi:type="string">email</item>
102                 <item name="source" xsi:type="string">email</item>
103                 <item name="dataScope" xsi:type="string">email</item>
104                 <item name="validation" xsi:type="array">
```

```
<field name="enable">
    <argument name="data" xsi:type="array">
        <item name="options" xsi:type="object">News\Career\Model\Page\Source\EnablingPin</item>
        <item name="label" xsi:type="string">translate="true">available</item>
        <item name="dataType" xsi:type="string">text</item>
        <item name="formElement" xsi:type="string">select</item>
        <item name="dataScope" xsi:type="string">available</item>
    </item>
</argument><settings>
<validation>
<rule name="required-entry" xsi:type="boolean">true</rule>
</validation>
</settings>
</field>

<field name="job_role">
    <argument name="data" xsi:type="array">
        <item name="config" xsi:type="array">
            <item name="datatype" xsi:type="string">text</item>
            <item name="label" xsi:type="string" translate="true">Job Role</item>
            <item name="formElement" xsi:type="string">input</item>
            <item name="source" xsi:type="string"> job_role</item>
            <item name="dataScope" xsi:type="string">job_role </item>
            <item name="validation" xsi:type="array">
                <item name="required-entry" xsi:type="boolean">true</item>
            </item>
        </item>
    </arguments>
</field>
<field name="place">
    <argument name="data" xsi:type="array">
        <item name="config" xsi:type="array">
            <item name="datatype" xsi:type="string">text</item>
            <item name="label" xsi:type="string" translate="true">Place</item>
            <item name="formElement" xsi:type="string">input</item>
            <item name="source" xsi:type="string"> place</item>
            <item name="dataScope" xsi:type="string"> place</item>
            <item name="validation" xsi:type="array">
                <item name="required-entry" xsi:type="boolean">true</item>
            </item>
        </item>
    </arguments>
</field>
<field name="email">
    <argument name="data" xsi:type="array">
        <item name="config" xsi:type="array">
            <item name="datatype" xsi:type="string">text</item>
            <item name="label" xsi:type="string" translate="true">Email</item>
            <item name="formElement" xsi:type="string">email</item>
            <item name="source" xsi:type="string">email</item>
            <item name="dataScope" xsi:type="string">email</item>
            <item name="validation" xsi:type="array">
```

Step 19: Create the career_sample_index.xml in view-adminhtml-layout folder: The index.xml file can define the layout of the blocks on the dashboard page, such as the position, size, and alignment of each block. It can also specify the content that should be displayed in each block.

app/code/News/Career/view/adminhtml/layout/career_sample_index.xml

```
News > Career > view > adminhtml > layout > career_sample_index.xml
1  <?xml version="1.0"?>
2  <!--
3  /**
4  * Copyright © Magento, Inc. All rights reserved.
5  * See COPYING.txt for license details.
6  */
7  -->
8  <page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page_configuration.xsd">
9      <body>
10         <referenceContainer name="content">
11             |     <uiComponent name="career_listing"/>
12         </referenceContainer>
13     </body>
14 </page>
15
```

Step 20: Create the career_sample_form.xml in view-adminhtml-layout folder:

The form.xml file can define the layout of the blocks on the dashboard page, such as the position, size, and alignment of each block. It can also specify the content that should be displayed in each block, such as Enabling options. The form.xml file include instructions for adding custom buttons or actions to the form page, or for modifying existing buttons or actions to add additional functionality or styling.

app/code/News/Career/view/adminhtml/layout/career_sample_form.xml

```
News > Career > view > adminhtml > layout > career_sample_form.xml
1  <?xml version="1.0"?>
2
3  <page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../../../../lib/internal/Magento/Framework/View/Layout/etc/page_configuration.xsd">
4      <update handle="styles"/>
5      <body>
6          <referenceContainer name="content">
7              |     <uiComponent name="career_form"/>
8          </referenceContainer>
9      </body>
10 </page>
11
```

2.2 Adding Buttons to the Form:

1. Back.php: In Magento the back button is used to navigate to the previous page in the users browsing history.

app/code/News/Career/Block/Adminhtml/Button/Back.php

News > Career > Block > Adminhtml > Button > Back.php

```

1 <?php
2
3 namespace News\Career\Block\Adminhtml\Button;
4
5 use Magento\Framework\View\Element\UiComponent\Control\ButtonProviderInterface;
6
7 class Back extends Generic implements ButtonProviderInterface
8 {
9     public function getButtonData()
10    {
11        return [
12            'label' => __('Back'),
13            'on_click' => sprintf("location.href = '%s';", $this->getBackUrl()),
14            'class' => 'back',
15            'sort_order' => 10,
16        ];
17    }
18    public function getBackUrl()
19    {
20        return $this->getUrl('*/*/');
21    }
22 }

```

2. Delete.php: The delete button is used to delete the details of the storelocator in which the user can add the other details in the store form.

app/code/News/Career/Block/Adminhtml/Button/Delete.php

```

News > Career > Block > Adminhtml > Button > Delete.php
1 <?php
2
3 namespace News\Career\Block\Adminhtml\Button;
4
5 //use Magento\Backend\Block\Widget\Context;
6 use Magento\Framework\View\Element\UiComponent\Control\ButtonProviderInterface;
7
8 class Delete extends Generic implements ButtonProviderInterface
9 {
10    // protected $context;
11
12    // public function __construct(
13    //     Context $context
14    // ) {
15    //     $this->context = $context;
16    // }
17
18    /**
19     * @return array
20     */
21
22    /**
23     * @inheritDoc
24     */
25
26    public function getButtonData()
27    {
28        $data = [];
29        $id = $this->context->getRequest()->getParam('id');
30        if ($id) {
31            $data = [
32                'label' => __('Delete'),
33                'class' => 'delete',
34                'on_click' => 'deleteConfirm(`' . __(
35                    'Are you sure you want to delete this file?',
36                    ) . '\n, \'' . $this->getDeleteUrl() . '\', {"data": {}})',
37                'sort_order' => 20,
38            ];
39        }
40        return $data;
41    }
42
43    /**
44     * * Url to send delete requests to.
45     *
46     * @return string
47     */
48    public function getDeleteUrl()
49    {
50        return $this->getUrl('*/*/');
51    }
52 }

```

3.Generic.php:

app/code/News/Career/Block/Adminhtml/Button/Generic.php

```
News > Career > Block > Adminhtml > Button > Generic.php
1 <?php
2
3 namespace News\Career\Block\Adminhtml\Button;
4
5 use Magento\Backend\Block\Widget\Context;
6 use Magento\CMS\Api\PageRepositoryInterface;
7
8 class Generic
9 {
10     protected $context;
11     protected $pageRepository;
12
13     public function __construct(
14         Context $context,
15         PageRepositoryInterface $pageRepository
16     ) {
17         $this->context = $context;
18         $this->pageRepository = $pageRepository;
19     }
20
21     public function getUrl($route = '', $params = [])
22     {
23         return $this->context->getUrlBuilder()->getUrl($route, $params);
24     }
25 }
```

4. Reset.php: It is typically used to undo any changes made to the form or configuration settings and restore the default values and found in the same page of save button in which the user can change various settings.

app/code/News/Career/Block/Adminhtml/Button/Reset.php

```
News > Career > Block > Adminhtml > Button > Reset.php
1 <?php
2
3 namespace News\Career\Block\Adminhtml\Button;
4
5 class Reset extends \Magento\Catalog\Block\Adminhtml\Product>Edit\Button\Generic
6 {
7     public function getButtonData()
8     {
9         return [
10             'label' => __('Reset'),
11             'class' => 'reset',
12             'on_click' => 'location.reload();',
13             'sort_order' => 30,
14         ];
15     }
16 }
```

5. Save.php: The save.php is used to apply the changes made to a form or configuration changes and update the corresponding entity in the system and the user can make changes.

app/code/News/Career/Block/Adminhtml/Button/Save.php

```

News > Career > Block > Adminhtml > Button > Save.php
1  <?php
2
3  namespace News\Career\Block\Adminhtml\Button;
4
5  use Magento\Framework\View\Element\UiComponent\Control\ButtonProviderInterface;
6  use Magento\Ui\Component\Control\Container;
7
8  class Save extends Generic implements ButtonProviderInterface
9  {
10     public function getButtonData()
11     {
12         return [
13             'label' => __('Save'),
14             'class' => 'save primary',
15             'data_attribute' => [
16                 'mage-init' => [
17                     'buttonAdapter' => [
18                         'actions' => [
19                             [
20                                 'targetName' => 'career_form.career_form',
21                                 'actionName' => 'save',
22                                 'params' => [
23                                     false,
24                                 ],
25                             ],
26                         ],
27                     ],
28                 ],
29             ],
30             'class_name' => Container::SPLIT_BUTTON,
31             'options' => $this->getOptions(),
32         ];
33     }
34
35     protected function getOptions()
36     {
37         $options[] = [
38             'id_hard' => 'save_and_new',
39             'label' => __('Save & New'),
40             'data_attribute' => [
41                 'mage-init' => [
42                     'buttonAdapter' => [
43                         'actions' => [
44                             [
45                                 'targetName' => 'career_form.career_form',
46                                 'actionName' => 'save',
47                                 'params' => [
48                                     true,
49                                     [
50                                         'back' => 'add',
51                                     ],
52                                 ],
53                             ],
54                         ],
55                     ],
56                 ],
57             ];
58         $options[] = [
59             'id_hard' => 'save_and_close',
60             'label' => __('Save & Close'),
61             'data_attribute' => [
62                 'mage-init' => [
63                     'buttonAdapter' => [
64                         'actions' => [
65                             [
66                                 'targetName' => 'career_form.career_form',
67                                 'actionName' => 'save',
68                                 'params' => [
69                                     true,
70                                 ],
71                             ],
72                         ],
73                     ],
74                 ],
75             ];
76     }
77 }

```

2.3 ADDING UI COMPONENT:

1. Delete.php: It is the standard file name used in the adminhtml controller directory to handle the deletions of specific entity in which the customer can delete the details.

app/code/News/Career/Controller/Adminhtml/Sample/Delete.php

```

News > Career > Controller > Adminhtml > Sample > *Delete.php
 1 <?php
 2 /**
 3  * Copyright © Magento, Inc. All rights reserved.
 4  * See COPYING.txt for license details.
 5  */
 6 namespace News\Career\Controller\Adminhtml\Sample;
 7
 8 use Magento\Framework\App\Action\HttpPostActionInterface;
 9
10 /**
11 * Delete career page action.
12 */
13 class Delete extends \Magento\Backend\App\Action implements HttpPostActionInterface
14 {
15     /**
16      * Authorization level of a basic admin session
17      *
18      * @see _isAllowed()
19      */
20     const ADMIN_RESOURCE = 'News_Career::sample_delete';
21
22     /**
23      * Delete action
24      *
25      * @return \Magento\Backend\Model\View\Result\Redirect
26      */
27     public function execute()
28     {
29         // check if we know what should be deleted
30         $id = $this->getRequest()->getParam('id');
31
32         /** @var \Magento\Backend\Model\View\Result\Redirect $resultRedirect */
33         $resultRedirect = $this->resultRedirectFactory->create();
34
35         if ($id) {
36             $title = '';
37             try {
38                 // init model and delete
39                 $model = $this->_objectManager->create(\News\Career\Model\Sample::class);
40                 $model->load($id);
41
42                 $title = $model->getTitle();
43                 $model->delete();
44
45                 // display success message
46                 $this->messageManager->addSuccessMessage(__('The page has been deleted.'));
47
48                 // go to grid
49                 $this->_eventManager->dispatch("adminhtml_sample_on_delete", [
50                     'title' => $title,
51                     'status' => 'success'
52                 ]);
53
54                 return $resultRedirect->setPath('*/*/*');
55             } catch (\Exception $e) {
56                 $this->_eventManager->dispatch(
57                     'adminhtml_sample_on_delete',
58                     ['title' => $title, 'status' => 'fail']
59                 );
60
61                 // display error message
62                 $this->messageManager->addErrorMassage($e->getMessage());
63                 // go back to edit form
64                 return $resultRedirect->setPath('*/*/edit', ['id' => $id]);
65             }
66         }
67
68         // display error message
69         $this->messageManager->addErrorMassage(__('We can't find a page to delete.'));
70
71         // go to grid

```

2. Edit.php: It is the standard file name used in the adminhtml controller directory to handle the deletions of specific entity in which the customer can edit the details.

app/code/News/Career/Controller/Adminhtml/Sample/Edit.php

```

use Magento\Backend\App\Action;
...
/** Edit career page action.
*/
class Edit extends \Magento\Backend\App\Action implements \HttpGetActionInterface
{
    /**
     * Authorization level of a basic admin session
     *
     * @see _isAllowed()
     */
    const ADMIN_RESOURCE = 'News_Career::save';

    /**
     * Core registry
     *
     * @var \Magento\Framework\Registry
     */
    protected $_coreRegistry;

    /**
     * @var \Magento\Framework\View\Result\PageFactory
     */
    protected $resultPageFactory;

    /**
     * @param Action\Context $context
     * @param \Magento\Framework\View\Result\PageFactory $resultPageFactory
     * @param \Magento\Framework\Registry $registry
     */
    public function __construct(
        Action\Context $context,
        \Magento\Framework\View\Result\PageFactory $resultPageFactory,
        \Magento\Framework\Registry $registry
    ) {
        $this->resultPageFactory = $resultPageFactory;
        $this->_coreRegistry = $registry;
        parent::__construct($context);
    }

    /**
     * Init actions
     *
     * @return \Magento\Backend\Model\View\Result\Page
     */
    protected function _initAction()
    {
        // load layout, set active menu and breadcrumbs
        /** @var \Magento\Backend\Model\View\Result\Page $resultPage */
        $resultPage = $this->resultPageFactory->create();
        $resultPage->setActiveMenu('News_Career::career_page')
            ->addBreadcrumb(__("Career"), __("Career"))
            ->addBreadcrumb(__("Manage Pages"), __("Manage Pages"));
        return $resultPage;
    }

    /**
     * Edit Career page.
     *
     * @return \Magento\Backend\Model\View\Result\Page|\Magento\Backend\Model\View\Result\Redirect
     * @SuppressWarnings(PHPMD.NPathComplexity)
     */
    public function execute()
    {
        // 1. Get ID and create model
        $id = $this->getRequest()->getParam('id');

        $model = $this->_objectManager->create(\News\Career\Model\Sample::class);

        // 2. Initial checking
        if ($id) {
            $model->load($id);
            if (!$model->getId()) {
                $this->messageManager->addErrorMessage(__('This page no longer exists.'));
                /** \Magento\Backend\Model\View\Result\Redirect $resultRedirect */
                $resultRedirect = $this->resultRedirectFactory->create();
                return $resultRedirect->setPath('*/*');
            }
        }

        // $this->_coreRegistry->register('cms_page', $model);

        // 5. Build edit form
        /** @var \Magento\Backend\Model\View\Result\Page $resultPage */
        $resultPage = $this->_initAction();
        $resultPage->addBreadcrumb(
            $id ? __("Edit Page") : __("New Page"),
            $id ? __("Edit Page") : __("New Page")
        );
        $resultPage->getConfig()->getTitle()->set($id ? __('Pages') :

```

2.4 UI COMPONENT:

1. PageActions.php: It is a file in the UI component directory that is responsible for handling various actions triggered by the user interface. This file receives the request from UI components and execute the corresponding action based on the request parameter.

app/code/News/Career/Ui/Component/Listing/Column/PageActions.php

News > Career > Ui > Component > Listing > Column > PageActions.php

```

1  <?php
2  /**
3   * Copyright © Magento, Inc. All rights reserved.
4   * See COPYING.txt for license details.
5   */
6  namespace News\Career\Ui\Component\Listing\Column;
7
8  use Magento\Cms\Block\Adminhtml\Page\Grid\Renderer\Action\UrlBuilder;
9  use Magento\Framework\App\ObjectManager;
10 use Magento\Framework\Escaper;
11 use Magento\Framework\UrlInterface;
12 use Magento\Framework\View\Element\UiComponent\ContextInterface;
13 use Magento\Framework\View\Element\UiComponentFactory;
14 use Magento\Ui\Component\Listing\Columns\Column;
15
16 /**
17 * Class prepare Page Actions
18 */
19 class PageActions extends Column
20 {
21     /** Url path */
22     const URL_PATH_EDIT = 'career/sample/edit';
23     const URL_PATH_DELETE = 'career/sample/delete';
24     /**
25      * @var \Magento\Cms\Block\Adminhtml\Page\Grid\Renderer\Action\UrlBuilder
26      */
27     protected $actionUrlBuilder;
28
29     /**
30      * @var \Magento\Framework\UrlInterface
31      */
32     protected $urlBuilder;
33
34     /**
35      * @var string
36      */
37     private $editUrl;
38
39     /**
40      * @var Escaper
41      */
42     private $escaper;
43
44     /**
45      * @param ContextInterface $context
46      * @param UiComponentFactory $uiComponentFactory
47      * @param UrlBuilder $actionUrlBuilder
48      * @param UrlInterface $urlBuilder
49      * @param array $components
50      * @param array $data
51      * @param string $editUrl
52      * @param \News\Career\ViewModel\Page\Grid\UrlBuilder|null $scopeUrlBuilder
53      */
54     public function __construct(
55         ContextInterface $context,
56         UiComponentFactory $uiComponentFactory,
57         UrlBuilder $actionUrlBuilder,
58         UrlInterface $urlBuilder,
59         array $components,
60         array $data,
61         $editUrl = self::URL_PATH_EDIT,
62         \News\Career\ViewModel\Page\Grid\UrlBuilder $scopeUrlBuilder = null
63     ) {
64         $this->urlBuilder = $urlBuilder;
65         $this->editUrl = $editUrl;
66         parent::__construct($context, $uiComponentFactory, $components, $data);
67         $this->scopeUrlBuilder = $scopeUrlBuilder ?: ObjectManager::getInstance()
68             ->get(\News\Career\ViewModel\Page\Grid\UrlBuilder::class);
69     }
70
71     /**
72      * @inheritDoc
73      */
74     public function prepareDataSource(array $dataSource)
75     {
76         if (isset($dataSource['data']['items'])) {
77             foreach ($dataSource['data']['items'] as &$item) {
78                 $name = $this->getData('name');
79                 if (isset($item['id'])) {
80                     $item[$name]['edit'] = [
81                         'href' => $this->urlBuilder->getUrl($this->editUrl, ['id' => $item['id']]),
82                         'label' => $this->escaper->escapeHtml($name)
83                     ];
84                 }
85             }
86         }
87     }
88 }

```

```

News > Career > Ui > Component > Listing > PageActions.php
  83
  84
  85
  86     $id = $this->getEscaper()->escapeHtml($item['id']);
  87     $item[$name]['delete'] = [
  88         'href' => $this->urlBuilder->getUrl(self::URL_PATH_DELETE, ["id" => $item['id']]),
  89         'label' => __("Delete"),
  90         'confirm' => [
  91             'id' => __("Delete %1", $id),
  92             'message' => __("Are you sure you want to delete a %1 th record?", $id),
  93         ],
  94         'post' => true,
  95     ];
  96 }
  97 if (isset($item['identifier'])) {
  98     $item[$name]['preview'] = [
  99         'href' => $this->scopeUrlBuilder->getUrl(
 100             $item['identifier'],
 101             isset($item['_first_store_id']) ? $item['_first_store_id'] : null,
 102             isset($item['store_code']) ? $item['store_code'] : null
 103         ),
 104         'label' => __("View"),
 105         'target' => '_blank'
 106     ];
 107 }
 108 }
 109
 110 return $dataSource;
 111
 112 /**
 113 * Get instance of escaper
 114 *
 115 * @return Escaper
 116 * @deprecated 101.0.7
 117 */
 118 private function getEscaper()
 119 {
 120     if (! $this->escaper) {
 121         $this->escaper = ObjectManager::getInstance()->get(Escaper::class);
 122     }
 123     return $this->escaper;
 124 }
 125
 126 }
 127

```

3. Front-End:

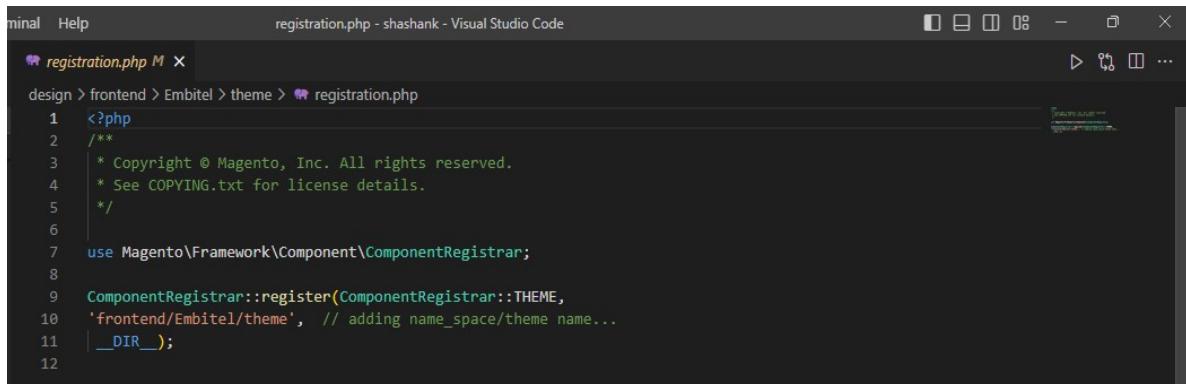
3.1 Creating Theme and Subtheme:

I utilised a custom theme and a child theme for this project, which inherit the parent theme's characteristics. The path of the theme is - app/design/frontend/Embitel/theme

app/design/frontend/Embitel/subtheme

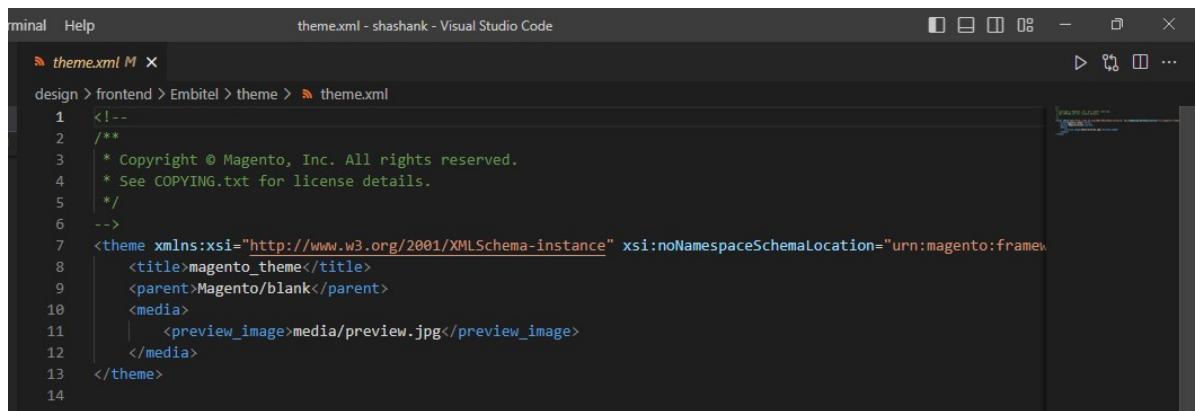
Below steps are followed while creating Theme:

- First I created registration.php for registering theme.



```
registration.php M - registration.php - shashank - Visual Studio Code
design > frontend > Embitel > theme > registration.php
1 <?php
2 /**
3  * Copyright © Magento, Inc. All rights reserved.
4  * See COPYING.txt for license details.
5 */
6
7 use Magento\Framework\Component\ComponentRegistrar;
8
9 ComponentRegistrar::register(ComponentRegistrar::THEME,
10 'frontend/Embitel/theme', // adding name_space/theme name...
11 | __DIR__);
12
```

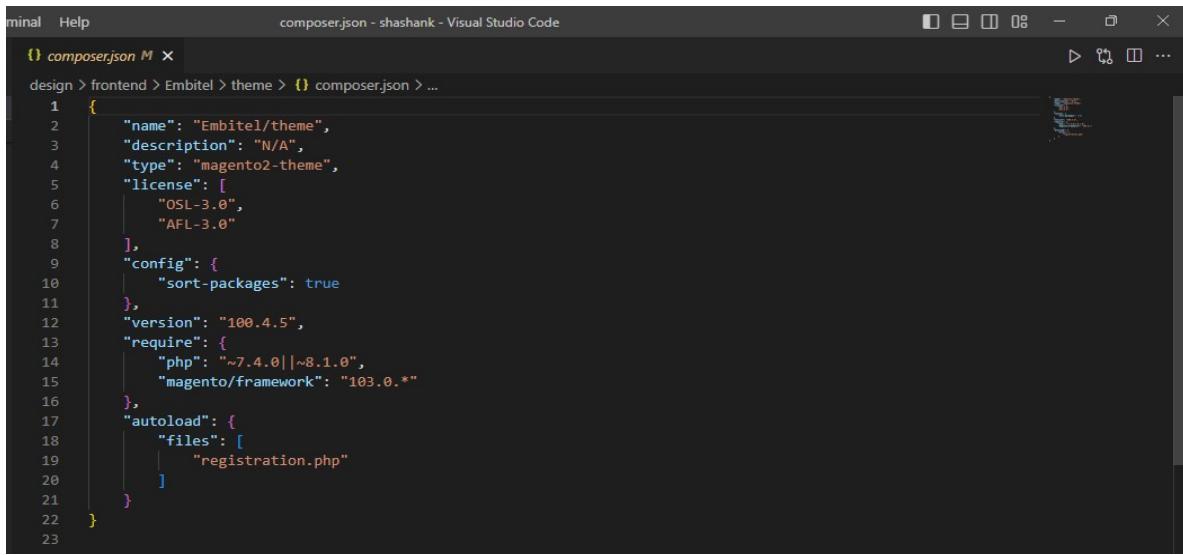
- Then created theme.xml for the theme



```
theme.xml M - theme.xml - shashank - Visual Studio Code
design > frontend > Embitel > theme > theme.xml
1 <!--
2 /**
3  * Copyright © Magento, Inc. All rights reserved.
4  * See COPYING.txt for license details.
5  */
6 -->
7 <theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:theme/etc/theme.xsd">
8     <title>magento_theme</title>
9     <parent>Magento/blank</parent>
10    <media>
11        <preview_image>media/preview.jpg</preview_image>
12    </media>
13 </theme>
14
```

In here parent theme for the Theme is magento's Blank theme and for the Subtheme's parent theme is Embitel theme which we created.

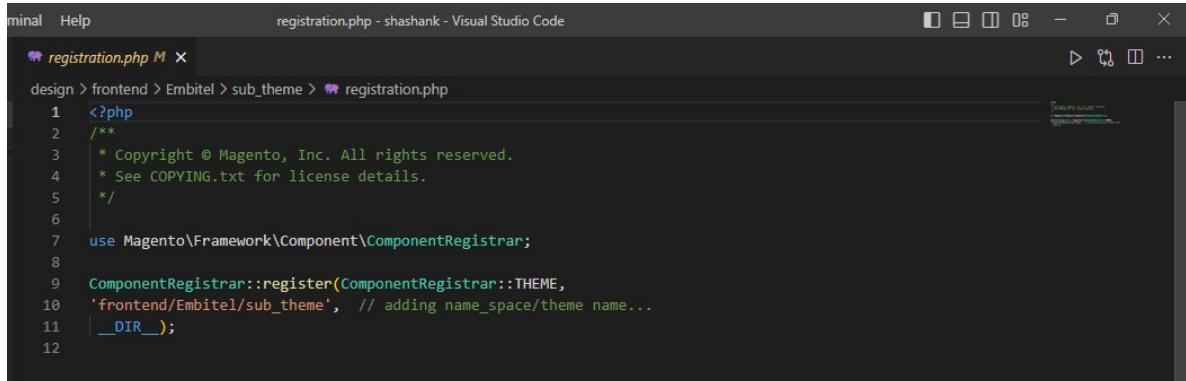
- Then created composer.json



```
terminal Help composer.json - shashank - Visual Studio Code
composer.json M
design > frontend > Embitel > theme > composer.json > ...
1 {
2     "name": "Embitel/theme",
3     "description": "N/A",
4     "type": "magento2-theme",
5     "license": [
6         "OSL-3.0",
7         "AFL-3.0"
8     ],
9     "config": {
10        "sort-packages": true
11    },
12    "version": "100.4.5",
13    "require": {
14        "php": "~7.4.0||~8.1.0",
15        "magento/framework": "103.0.*"
16    },
17    "autoload": {
18        "files": [
19            "registration.php"
20        ]
21    }
22 }
23
```

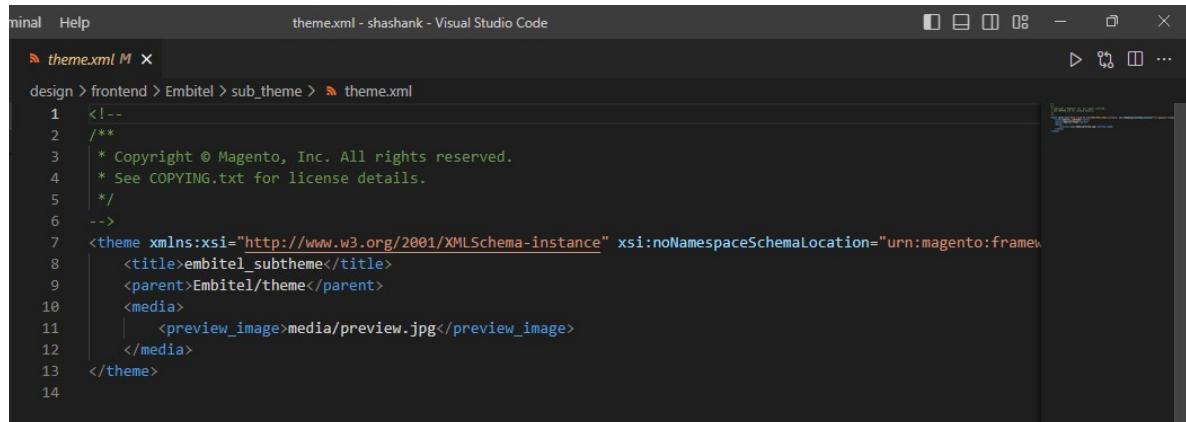
Creating a subtheme

- Creating registration.php



```
terminal Help registration.php - shashank - Visual Studio Code
registration.php M
design > frontend > Embitel > sub_theme > registration.php
1 <?php
2 /**
3  * Copyright © Magento, Inc. All rights reserved.
4  * See COPYING.txt for license details.
5  */
6
7 use Magento\Framework\Component\ComponentRegistrar;
8
9 ComponentRegistrar::register(ComponentRegistrar::THEME,
10     'frontend/Embitel/sub_theme', // adding name_space/theme name...
11     __DIR__);
12
```

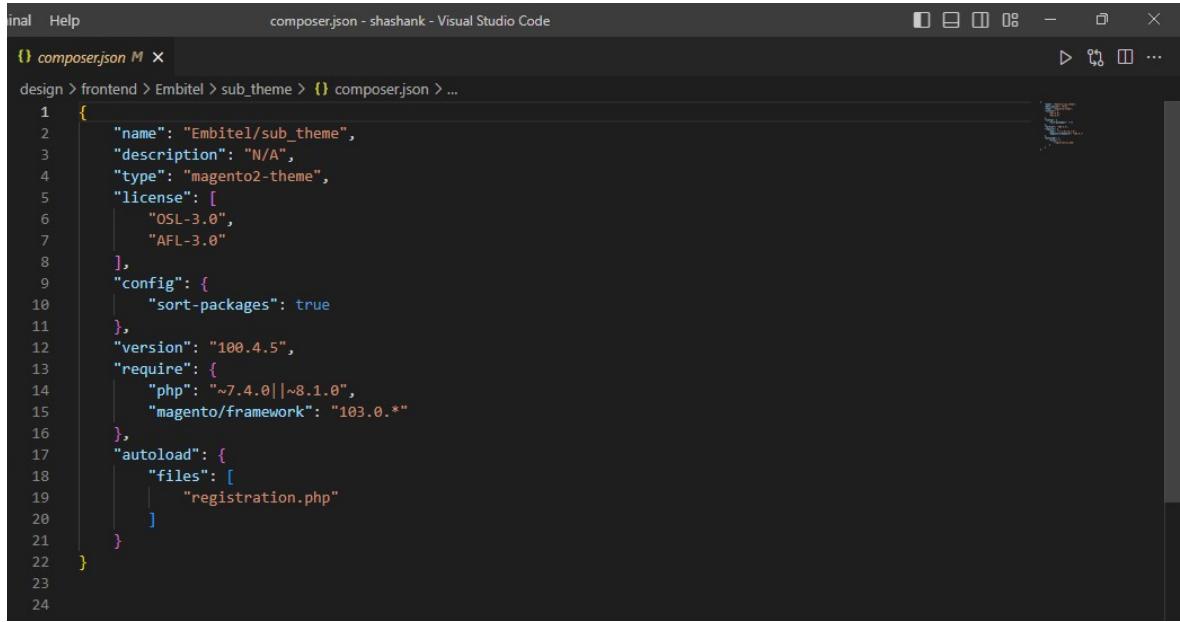
- Creating theme.xml



```
terminal Help theme.xml - shashank - Visual Studio Code
theme.xml M
design > frontend > Embitel > sub_theme > theme.xml
1 <!--
2 /**
3  * Copyright © Magento, Inc. All rights reserved.
4  * See COPYING.txt for license details.
5  */
6 -->
7 <theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Theme/etc/theme.xsd">
8     <title>embitel_subtheme</title>
9     <parent>Embitel/theme</parent>
10    <media>
11        <preview_image>media/preview.jpg</preview_image>
12    </media>
13 </theme>
14
```

In here parent theme for the Theme is Embitel

➤ Creating composer.json



The screenshot shows a Visual Studio Code window with the title "composer.json - shashank - Visual Studio Code". The code editor displays the following JSON content:

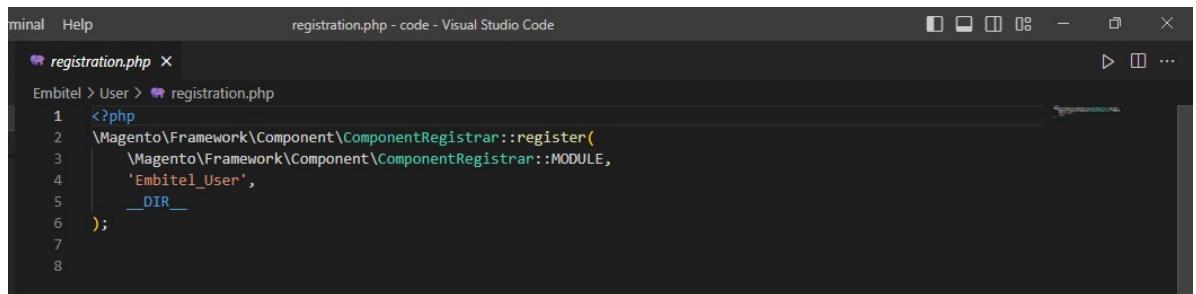
```
1  {
2      "name": "Embitel/sub_theme",
3      "description": "N/A",
4      "type": "magento2-theme",
5      "license": [
6          "OSL-3.0",
7          "AFL-3.0"
8      ],
9      "config": {
10         "sort-packages": true
11     },
12     "version": "100.4.5",
13     "require": {
14         "php": "~7.4.0||~8.1.0",
15         "magento/framework": "103.0.*"
16     },
17     "autoload": {
18         "files": [
19             "registration.php"
20         ]
21     }
22 }
23
24
```

Creating Module:

In this project we have created a module called “User” in the “Embitel” namespace.

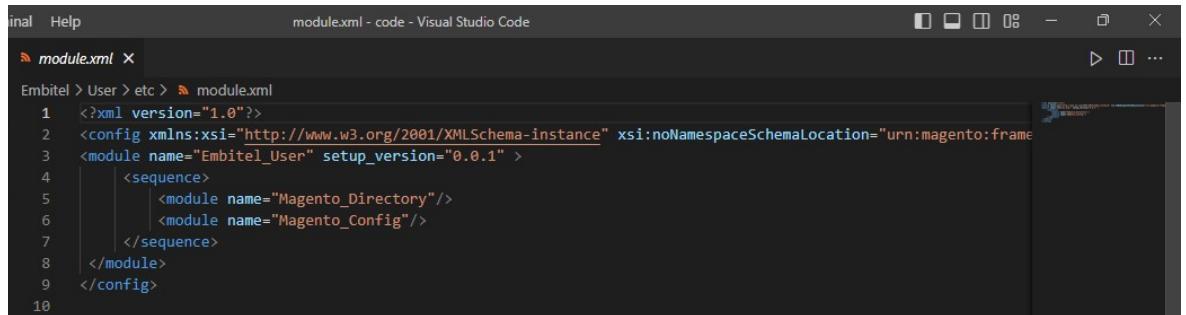
Below steps are used for create a module –

- First we created registration.php for registering the module.



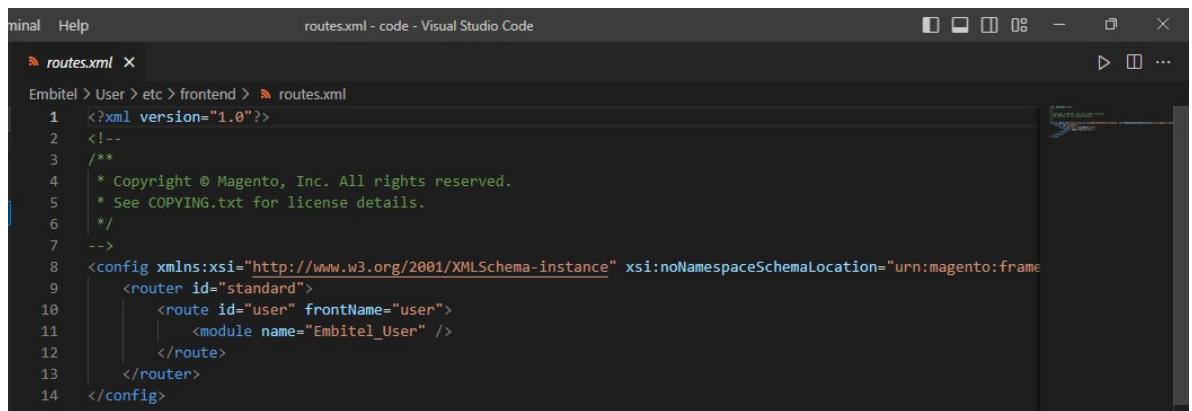
```
registration.php
1 <?php
2 \Magento\Framework\Component\ComponentRegistrar::register(
3     \Magento\Framework\Component\ComponentRegistrar::MODULE,
4     'Embitel_User',
5     __DIR__
6 );
7
8
```

- Then created *module.xml* inside *etc* folder- etc/module.xml



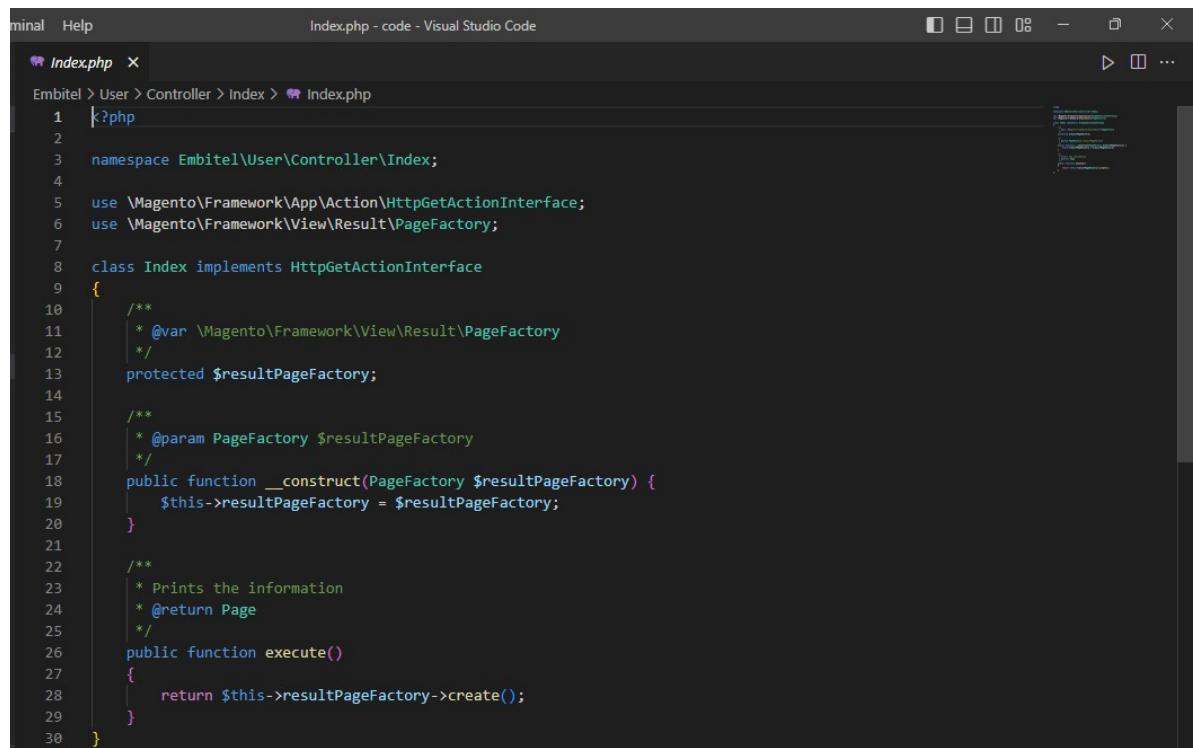
```
module.xml
1 <?xml version="1.0"?>
2 <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Module/etc/module.xsd">
3     <module name="Embitel_User" setup_version="0.0.1" >
4         <sequence>
5             <module name="Magento_Directory"/>
6             <module name="Magento_Config"/>
7         </sequence>
8     </module>
9 
```

- Then we have to create the router for that in the we have to create *routes.xml* in the *etc* folder – etc/frontend/routes.xml



```
routes.xml
1 <?xml version="1.0"?>
2 <!--
3 /**
4  * Copyright © Magento, Inc. All rights reserved.
5  * See COPYING.txt for license details.
6  */
7 -->
8 <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:Router/etc/routes.xsd">
9     <router id="standard">
10         <route id="user" frontName="user">
11             <module name="Embitel_User" />
12         </route>
13     </router>
14 
```

- After *routes.xml* we have to control for controlling the routes inside the *Controller* folder – Controller/index/Index.php

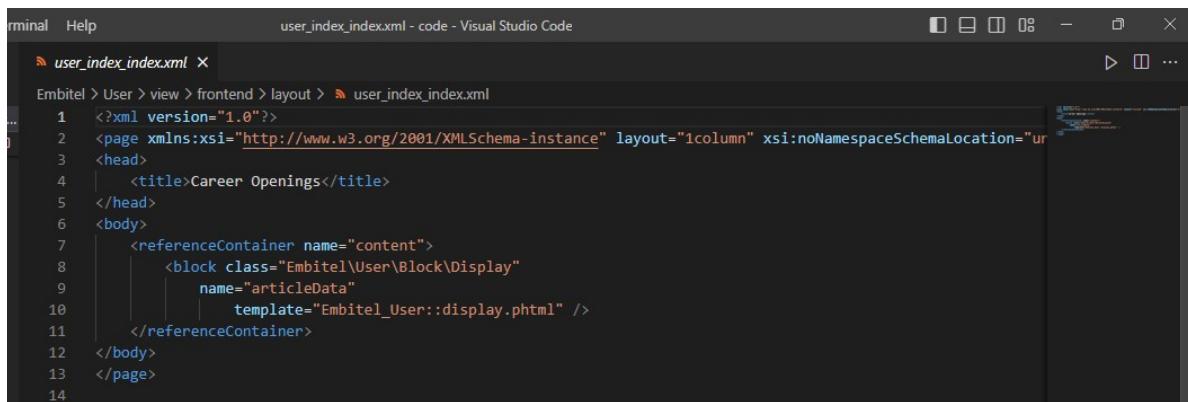


```

1 <?php
2
3 namespace Embitel\User\Controller\Index;
4
5 use \Magento\Framework\App\Action\HttpGetActionInterface;
6 use \Magento\Framework\View\Result\PageFactory;
7
8 class Index implements HttpGetActionInterface
9 {
10
11     /**
12      * @var \Magento\Framework\View\Result\PageFactory
13      */
14     protected $resultPageFactory;
15
16     /**
17      * @param PageFactory $resultPageFactory
18      */
19     public function __construct(PageFactory $resultPageFactory) {
20         $this->resultPageFactory = $resultPageFactory;
21     }
22
23     /**
24      * Prints the information
25      * @return Page
26      */
27     public function execute()
28     {
29         return $this->resultPageFactory->create();
30     }
}

```

- we have to create layout file named as moduleName_folder_controllerFileName.xml, in here User is our module and inside controller index is the folder inside that index.php is the controller file. So file name is user_index_index.xml

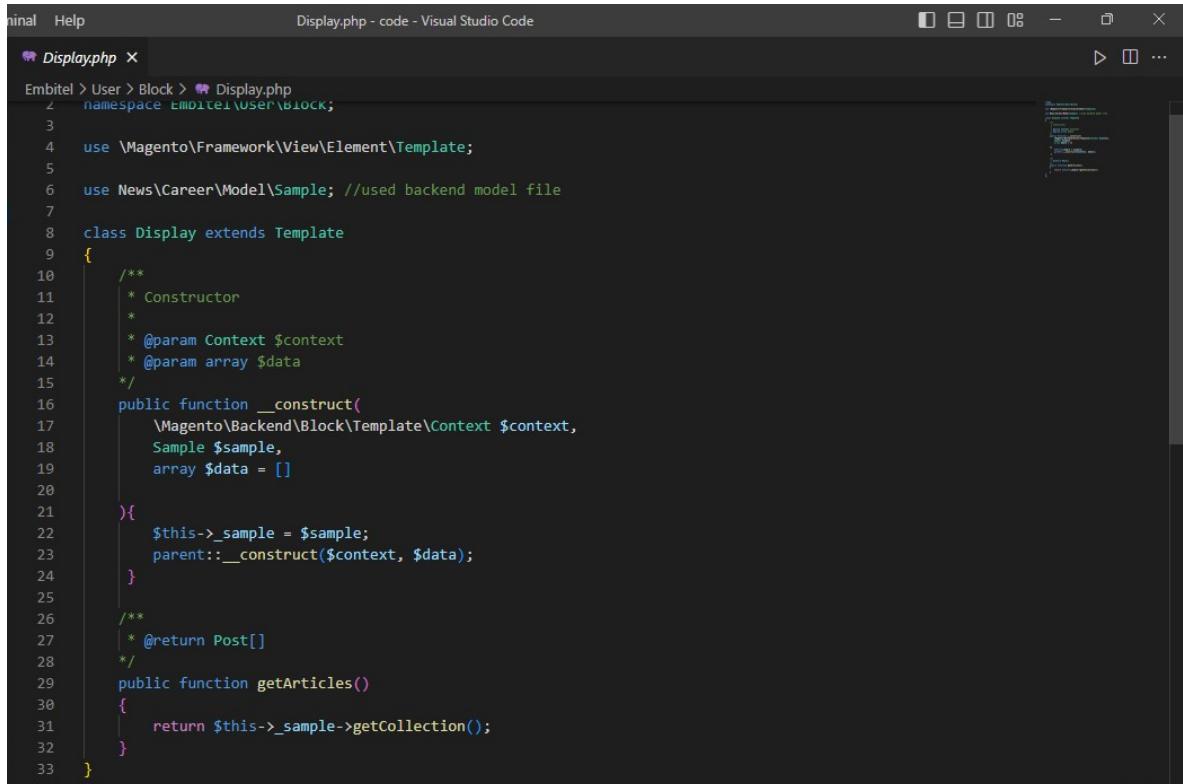


```

1 <?xml version="1.0"?>
2 <page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" layout="1column" xsi:noNamespaceSchemaLocation="urn:xsd:magento:page-layout:1.0.xsd">
3     <head>
4         <title>Career Openings</title>
5     </head>
6     <body>
7         <referenceContainer name="content">
8             <block class="Embtel\User\Block\Display"
9                   name="articleData"
10                  template="Embtel_User::display.phtml" />
11         </referenceContainer>
12     </body>
13 </page>
14

```

- Here I have called block class that is Display.php which resides inside Embitel/User/Block

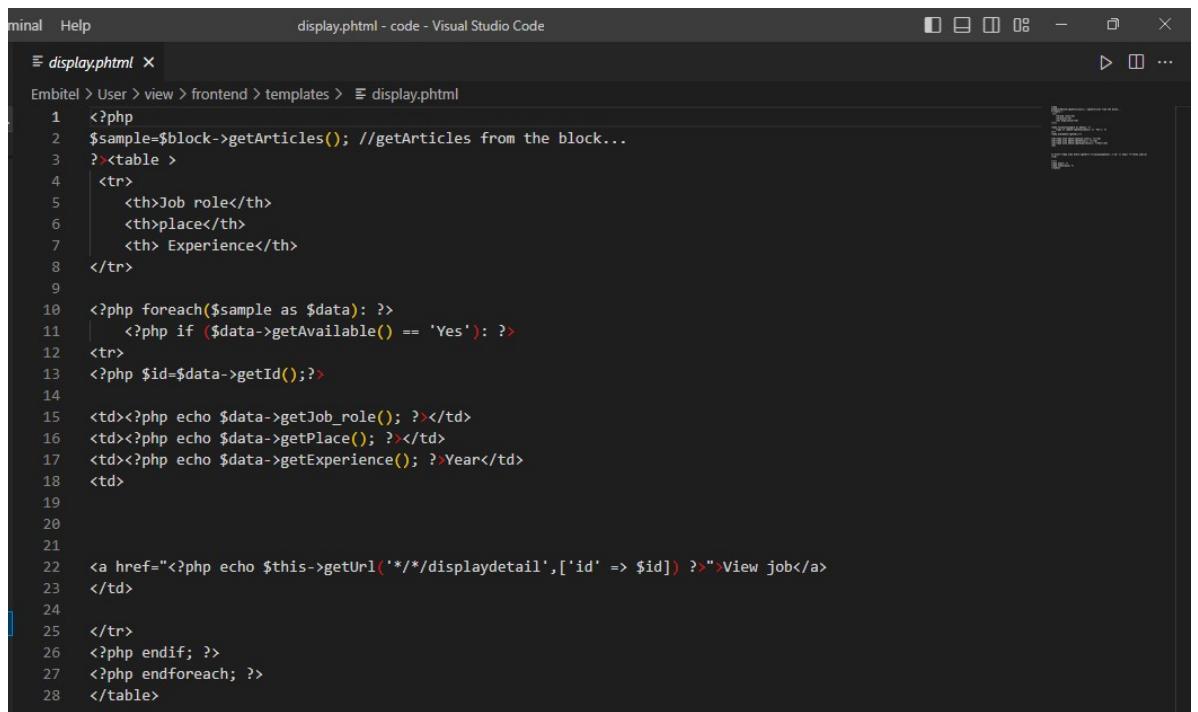


```

  terminal  Help
Display.php - code - Visual Studio Code
  ↻ Display.php ×
Embitel > User > Block > Display.php
  ↵ namespace EMBITEL\User\Block;
  ↵
  ↵ use \Magento\Framework\View\Element\Template;
  ↵
  ↵ use News\Career\Model\Sample; //used backend model file
  ↵
  ↵ class Display extends Template
  ↵ {
  ↵     /**
  ↵      * Constructor
  ↵      *
  ↵      * @param Context $context
  ↵      * @param array $data
  ↵      */
  ↵     public function __construct(
  ↵         \Magento\Backend\Block\Template\Context $context,
  ↵         Sample $sample,
  ↵         array $data = []
  ↵     ){
  ↵         $this->_sample = $sample;
  ↵         parent::__construct($context, $data);
  ↵     }
  ↵
  ↵     /**
  ↵      * @return Post[]
  ↵      */
  ↵     public function getArticles()
  ↵     {
  ↵         return $this->_sample->getCollection();
  ↵     }
  ↵ }

```

- After we have to create the *display.phtml* file inside the *view/frontend/templates/display.phtml*

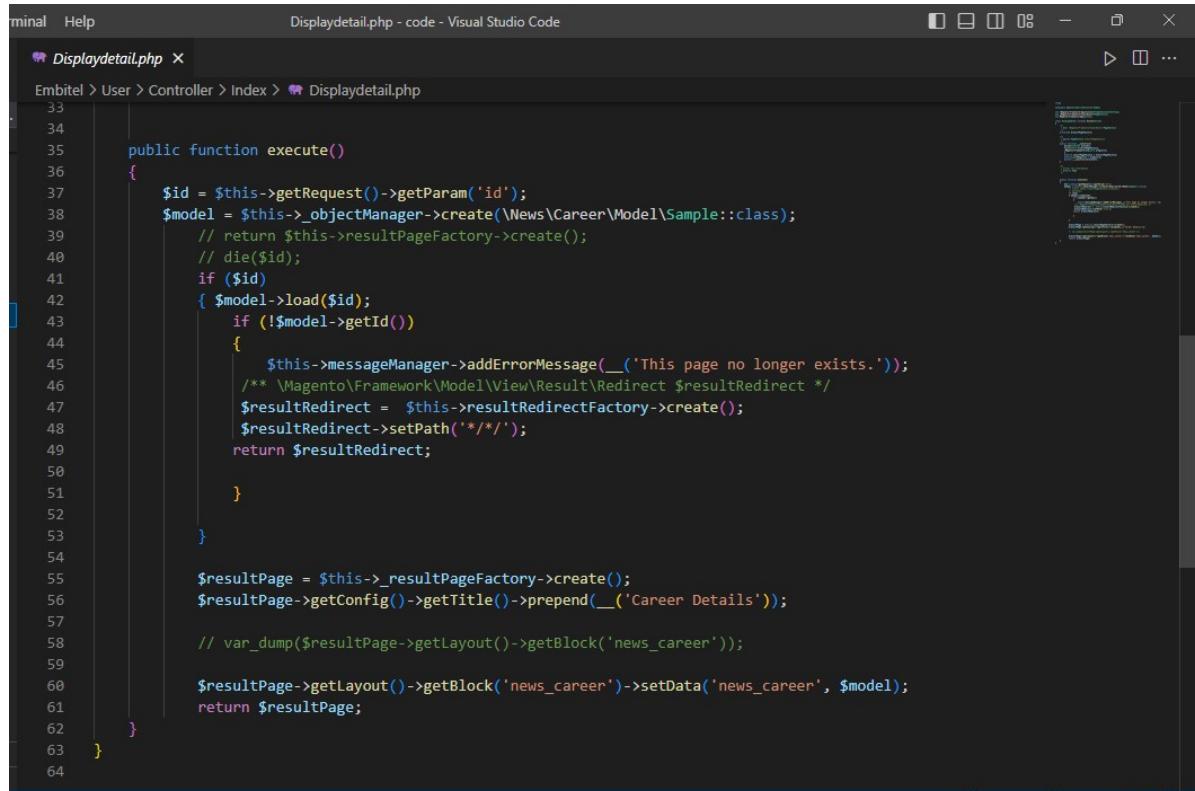


```

  terminal  Help
display.phtml - code - Visual Studio Code
  ↻ display.phtml ×
Embitel > User > view > frontend > templates > display.phtml
  ↵ <?php
  ↵ $sample=$block->getArticles(); //getArticles from the block...
  ↵ ?><table>
  ↵ <tr>
  ↵   <th>Job role</th>
  ↵   <th>place</th>
  ↵   <th> Experience</th>
  ↵ </tr>
  ↵
  ↵ <?php foreach($sample as $data): ?>
  ↵   <?php if ($data->getAvailable() == 'Yes'): ?>
  ↵   <tr>
  ↵     <?php $id=$data->getId();?>
  ↵
  ↵     <td><?php echo $data->getJob_role(); ?></td>
  ↵     <td><?php echo $data->getPlace(); ?></td>
  ↵     <td><?php echo $data->getExperience(); ?>Year</td>
  ↵   </td>
  ↵
  ↵   <a href="<?php echo $this->getUrl('*/*displaydetail',['id' => $id]) ?>">View job</a>
  ↵ </td>
  ↵
  ↵ </tr>
  ↵ <?php endif; ?>
  ↵ <?php endforeach; ?>
  ↵ </table>

```

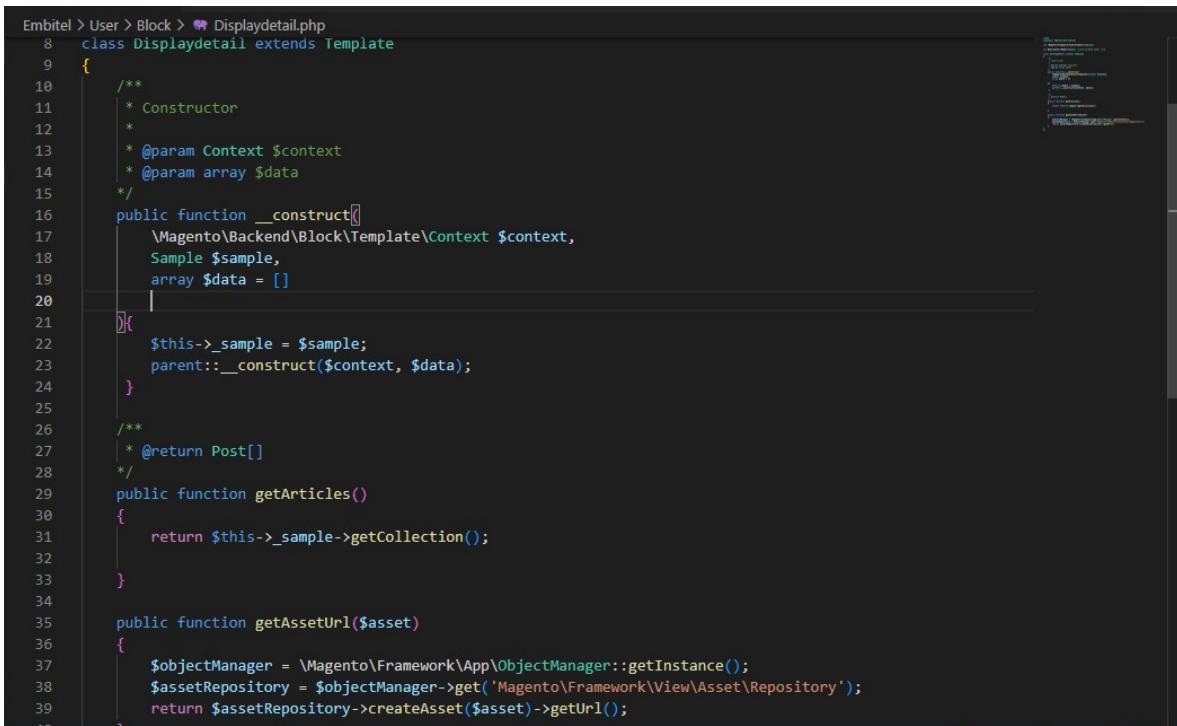
- in display.phtml file we are displaying the available jobs, if they are interested they can view and apply to the job in next page
- To redirect to next page I have called displaydetail controller class in the above mentioned file
- Displaydetail.php controller will display the detailed display of the job and it provides application form in a same page



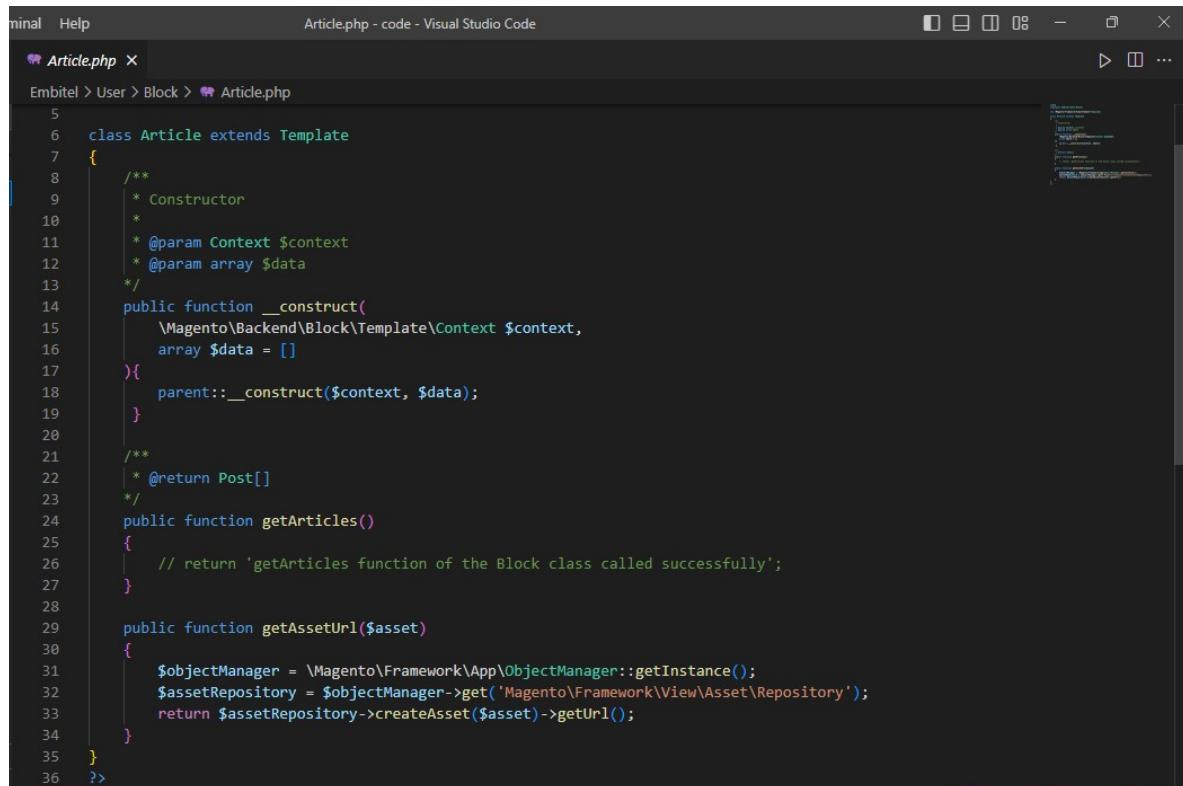
The screenshot shows a Visual Studio Code window with the title "Displaydetail.php - code - Visual Studio Code". The file path is "Embitel > User > Controller > Index > Displaydetail.php". The code is written in PHP and defines a public function execute(). It handles an ID parameter, creates a model object, and checks if it exists. If not, it adds an error message and creates a result redirect. Otherwise, it creates a result page, prepends a career details title, and sets the news_career block data. The code is color-coded for syntax.

```
33
34
35     public function execute()
36     {
37         $id = $this->getRequest()->getParam('id');
38         $model = $this->objectManager->create(\News\Career\Model\Sample::class);
39         // return $this->resultPageFactory->create();
40         // die($id);
41         if ($id)
42         {
43             $model->load($id);
44             if (!$model->getId())
45             {
46                 $this->messageManager->addErrorMessage(__('This page no longer exists.'));
47                 /** \Magento\Framework\Model\View\Result\Redirect $resultRedirect */
48                 $resultRedirect = $this->resultRedirectFactory->create();
49                 $resultRedirect->setPath('*/*');
50                 return $resultRedirect;
51             }
52         }
53
54         $resultPage = $this->resultPageFactory->create();
55         $resultPage->getConfig()->getTitle()->prepend(__('Career Details'));
56
57         // var_dump($resultPage->getLayout()->getBlock('news_career'));
58
59         $resultPage->getLayout()->getBlock('news_career')->setData('news_career', $model);
60
61         return $resultPage;
62     }
63 }
64 }
```

- To display application form and job details I created two block classes Article.php and Displaydetail.php
- In block folder display.php and displaydetail.php I have used
- *News/Career/Model/Sample* to access the model file

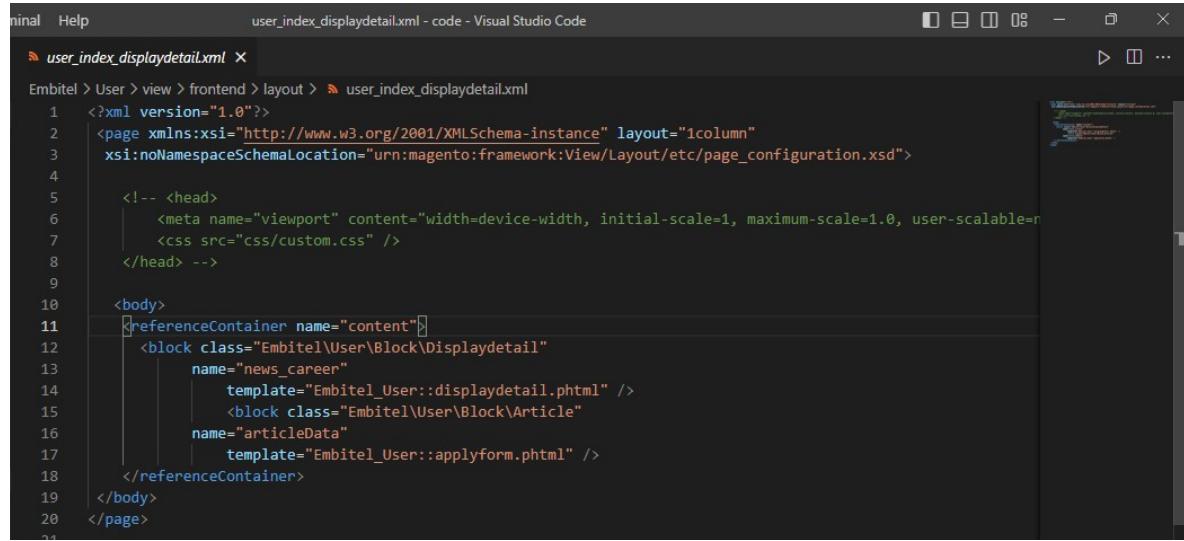


```
Embitel > User > Block > Article.php
8  class Displaydetail extends Template
9  {
10  /**
11  * Constructor
12  *
13  * @param Context $context
14  * @param array $data
15  */
16  public function __construct()
17  \Magento\Backend\Block\Template\Context $context,
18  Sample $sample,
19  array $data = []
20  |
21  [
22  $this->_sample = $sample;
23  parent::__construct($context, $data);
24  ]
25  /**
26  * @return Post[]
27  */
28  public function getArticles()
29  {
30  return $this->_sample->getCollection();
31  }
32  }
33  /**
34  * @return Asset[]
35  */
36  public function getAssetUrl($asset)
37  {
38  $objectManager = \Magento\Framework\App\ObjectManager::getInstance();
39  $assetRepository = $objectManager->get('Magento\Framework\View\Asset\Repository');
40  return $assetRepository->createAsset($asset)->getUrl();
41  }
```

➤ *Article.php*

```
Embitel > User > Block > Article.php
5
6  class Article extends Template
7  {
8  /**
9  * Constructor
10  *
11  * @param Context $context
12  * @param array $data
13  */
14  public function __construct(
15  \Magento\Backend\Block\Template\Context $context,
16  array $data = []
17  ){
18  parent::__construct($context, $data);
19  }
20  /**
21  * @return Post[]
22  */
23  public function getArticles()
24  {
25  // return 'getArticles function of the Block class called successfully';
26  }
27  /**
28  * @return Asset[]
29  */
30  public function getAssetUrl($asset)
31  {
32  $objectManager = \Magento\Framework\App\ObjectManager::getInstance();
33  $assetRepository = $objectManager->get('Magento\Framework\View\Asset\Repository');
34  return $assetRepository->createAsset($asset)->getUrl();
35  }
36  ?>
```

- Layout file user_index_displaydetail.xml here I added two block class inside reference container two display job description and application form



```

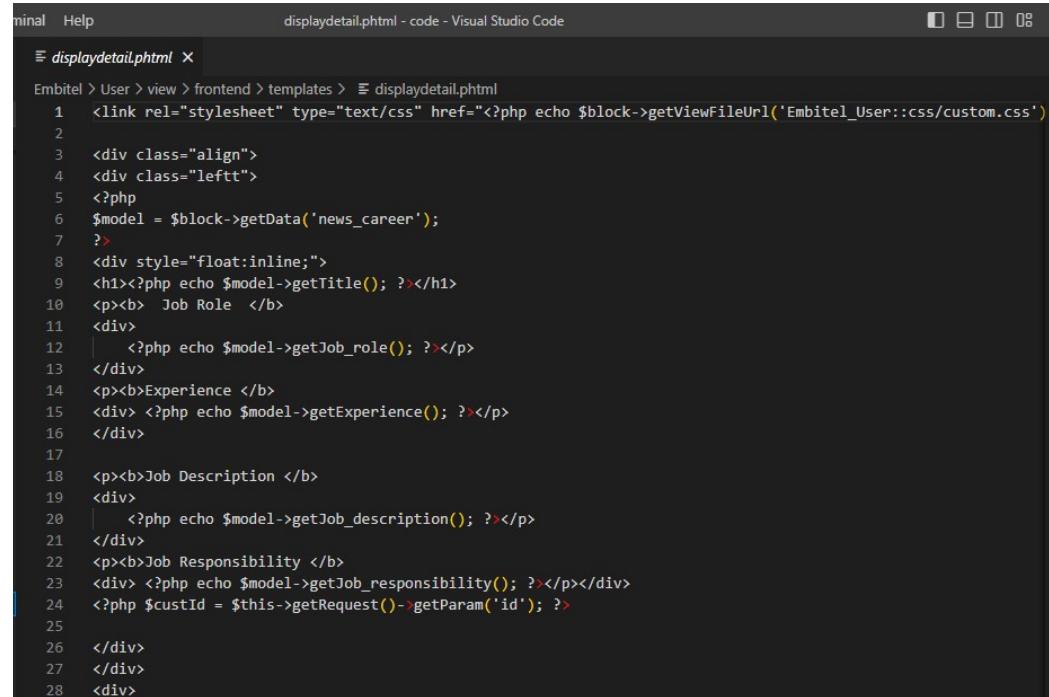
<?xml version="1.0"?>
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" layout="1column"
xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page_configuration.xsd">

<!-- <head>
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0, user-scalable=no" />
    <css src="css/custom.css" />
</head> -->

<body>
<referenceContainer name="content">
    <block class="Embitel\User\Block\Displaydetail"
        name="news_career"
        template="Embitel_User::displaydetail.phtml" />
    <block class="Embitel\User\Block\Article"
        name="articleData"
        template="Embitel_User::applyform.phtml" />
</referenceContainer>
</body>
</page>

```

- Two templates are called in the above layout file
- Displaydetail.phtml and apply.phtml

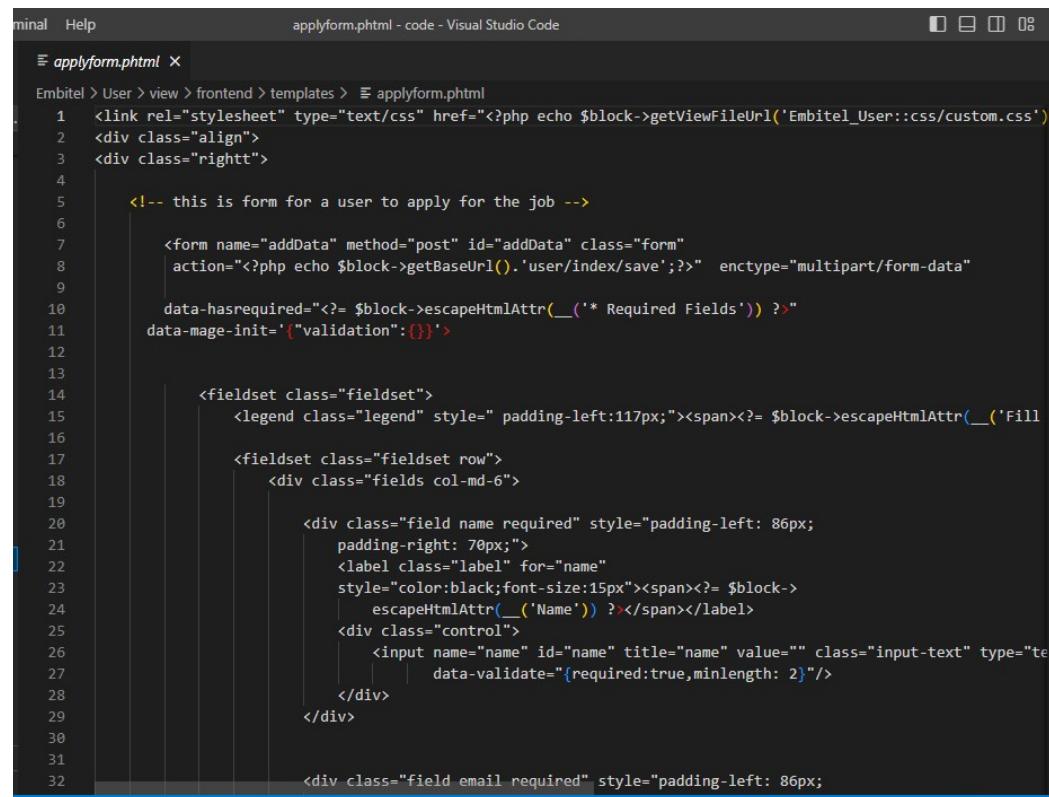


```

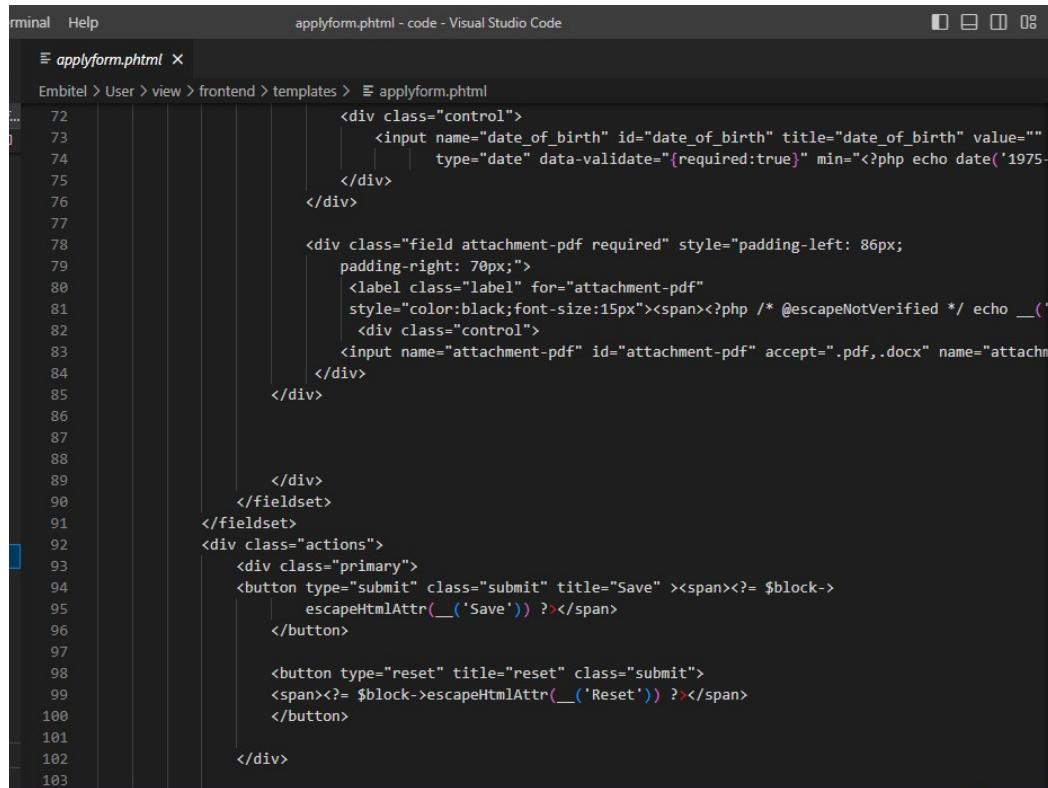
<link rel="stylesheet" type="text/css" href=<?php echo $block->getViewFileUrl('Embitel_User::css/custom.css')?>

<div class="align">
<div class="leftt">
<?php
$model = $block->getData('news_career');
?>
<div style="float:inline;">
<h1><?php echo $model->getTitle(); ?></h1>
<p><b> Job Role </b></p>
<div>
    <?php echo $model->getJob_role(); ?>
</div>
<p><b>Experience </b></p>
<div> <?php echo $model->getExperience(); ?></div>
</div>
<p><b>Job Description </b></p>
<div>
    <?php echo $model->getJob_description(); ?>
</div>
<p><b>Job Responsibility </b></p>
<div> <?php echo $model->getJob_responsibility(); ?></div>
<?php $custId = $this->getRequest()->getParam('id'); ?>
</div>
</div>
</div>

```

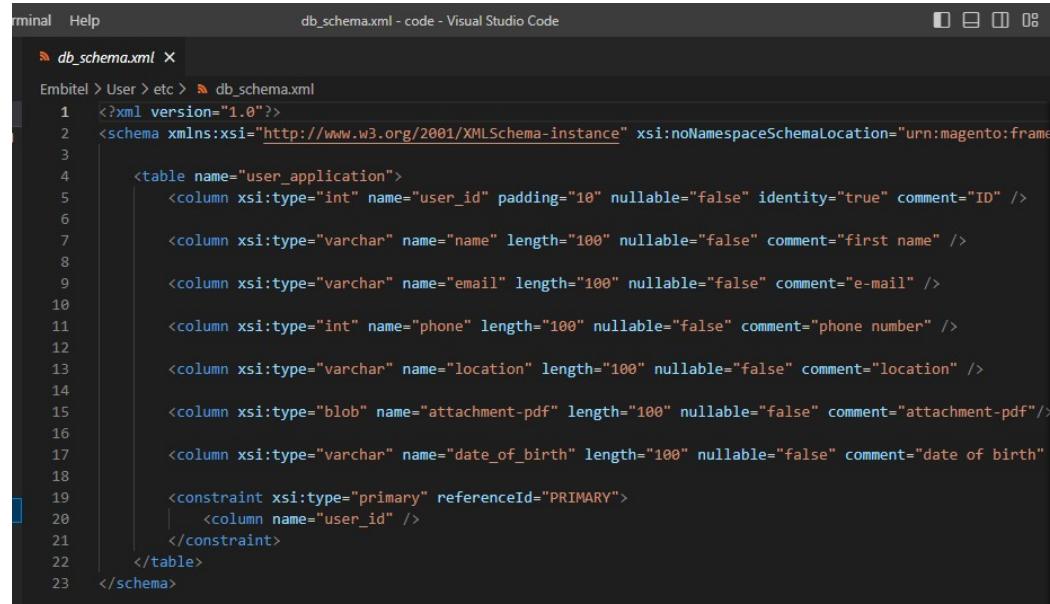


```
terminal Help applyform.phtml ×
Embitel > User > view > frontend > templates > applyform.phtml
1  <link rel="stylesheet" type="text/css" href="= $block-&gt;getViewFileUrl('Embitel_User::css/custom.css') ?"
2  <div class="align">
3  <div class="rightt">
4
5      <!-- this is form for a user to apply for the job -->
6
7      <form name="addData" method="post" id="addData" class="form"
8          action="= $block-&gt;getBaseUrl() . 'user/index/save'; ?" enctype="multipart/form-data"
9
10         data-hasrequired="= $block-&gt;escapeHtmlAttr(__('Required Fields')) ?&gt;" data-mage-init='{"validation":{} }'
11
12
13
14         &lt;fieldset class="fieldset"&gt;
15             &lt;legend class="legend" style="padding-left:117px;"&gt;&lt;span&gt;<?= $block-&gt;escapeHtmlAttr(__('Fill
16
17             &lt;fieldset class="fieldset row"&gt;
18                 &lt;div class="fields col-md-6"&gt;
19
20                     &lt;div class="field name required" style="padding-left: 86px;
21                         padding-right: 70px;"&gt;
22                         &lt;label class="label" for="name"
23                             style="color:black;font-size:15px"&gt;&lt;span&gt;<?= $block-&gt;
24                             escapeHtmlAttr(__('Name')) ?&gt;&lt;/span&gt;&lt;/label&gt;
25                         &lt;div class="control"&gt;
26                             &lt;input name="name" id="name" title="name" value="" class="input-text" type="text"
27                                 data-validate="{required:true,minlength: 2}"/&gt;
28                         &lt;/div&gt;
29                     &lt;/div&gt;
30
31
32             &lt;div class="field email required" style="padding-left: 86px;</pre
```



```
terminal Help applyform.phtml ×
Embitel > User > view > frontend > templates > applyform.phtml
...
72             <div class="control">
73                 <input name="date_of_birth" id="date_of_birth" title="date_of_birth" value=""
74                     type="date" data-validate="{required:true}" min="= $block-&gt;date('1975-
75
76             &lt;/div&gt;
77
78             &lt;div class="field attachment-pdf required" style="padding-left: 86px;
79                 padding-right: 70px;"&gt;
80                 &lt;label class="label" for="attachment-pdf"
81                     style="color:black;font-size:15px"&gt;&lt;span&gt;@escapeNotVerified / echo __('
82                     &lt;div class="control"&gt;
83                         &lt;input name="attachment-pdf" id="attachment-pdf" accept=".pdf,.docx" name="attachm
84                     &lt;/div&gt;
85                 &lt;/div&gt;
86
87
88             &lt;/div&gt;
89         &lt;/fieldset&gt;
90     &lt;/fieldset&gt;
91     &lt;div class="actions"&gt;
92         &lt;div class="primary"&gt;
93             &lt;button type="submit" class="submit" title="Save" &gt;&lt;span&gt;<?= $block-&gt;
94                 escapeHtmlAttr(__('Save')) ?&gt;&lt;/span&gt;
95             &lt;/button&gt;
96
97             &lt;button type="reset" title="reset" class="submit"&gt;
98                 &lt;span&gt;<?= $block-&gt;escapeHtmlAttr(__('Reset')) ?&gt;&lt;/span&gt;
99             &lt;/button&gt;
100
101         &lt;/div&gt;
102     &lt;/div&gt;</pre
```

- When user enters the form field it should be stored in DB so I created db_schema inside etc folder



```

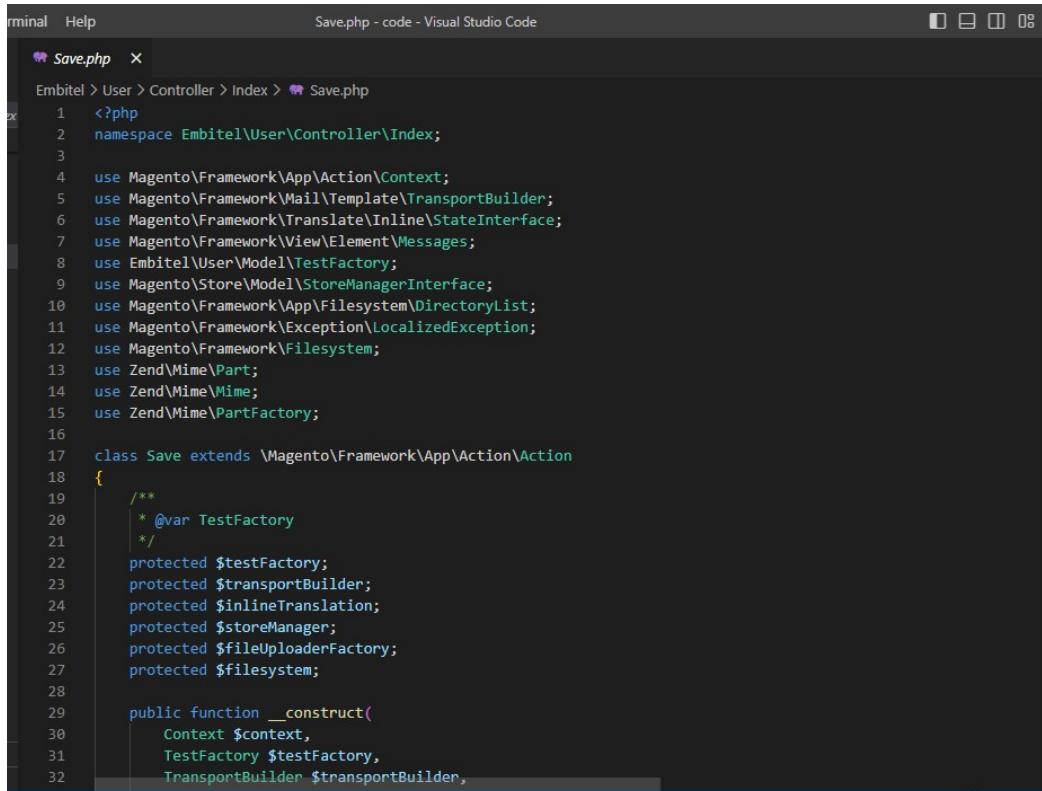
Terminal Help
db_schema.xml - code - Visual Studio Code

db_schema.xml ×

Embitel > User > etc > db_schema.xml
1  <?xml version="1.0"?>
2  <schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework
3
4      <table name="user_application">
5          <column xsi:type="int" name="user_id" padding="10" nullable="false" identity="true" comment="ID" />
6
7          <column xsi:type="varchar" name="name" length="100" nullable="false" comment="first name" />
8
9          <column xsi:type="varchar" name="email" length="100" nullable="false" comment="e-mail" />
10
11         <column xsi:type="int" name="phone" length="100" nullable="false" comment="phone number" />
12
13         <column xsi:type="varchar" name="location" length="100" nullable="false" comment="location" />
14
15         <column xsi:type="blob" name="attachment-pdf" length="100" nullable="false" comment="attachment-pdf"/>
16
17         <column xsi:type="varchar" name="date_of_birth" length="100" nullable="false" comment="date of birth" />
18
19         <constraint xsi:type="primary" referenceId="PRIMARY">
20             <column name="user_id" />
21         </constraint>
22     </table>
23 </schema>

```

- To store the data in database I created Save.php in controller and here I am sending mail from applicant to admin.
- In mail attaching the resume also, so in save.php we have written logic for sending mail also



```

Terminal Help
Save.php - code - Visual Studio Code

Save.php ×

Embitel > User > Controller > Index > Save.php
1  <?php
2  namespace Embitel\User\Controller\Index;
3
4  use Magento\Framework\App\Action\Context;
5  use Magento\Framework\Mail\Template\TransportBuilder;
6  use Magento\Framework\Translate\Inline\StateInterface;
7  use Magento\Framework\View\Element\Messages;
8  use Embitel\User\Model\TestFactory;
9  use Magento\Store\Model\StoreManagerInterface;
10 use Magento\Framework\App\Filesystem\DirectoryList;
11 use Magento\Framework\Exception\LocalizedException;
12 use Magento\Framework\Filesystem;
13 use Zend\Mime\Part;
14 use Zend\Mime\Mime;
15 use Zend\Mime\PartFactory;
16
17 class Save extends \Magento\Framework\App\Action\Action
18 {
19
    /**
     * @var TestFactory
     */
20
21    protected $testFactory;
22    protected $transportBuilder;
23    protected $inlineTranslation;
24    protected $storeManager;
25    protected $fileUploaderFactory;
26    protected $filesystem;
27
28
29    public function __construct(
30        Context $context,
31        TestFactory $testFactory,
32        TransportBuilder $transportBuilder,

```

Save.php - code - Visual Studio Code

```

  terminal  Help
  Save.php  ×
  Embitel > User > Controller > Index > Save.php
  34     StoreManagerInterface $storeManager,
  35     \Magento\MediaStorage\Model\File\UploaderFactory $fileUploaderFactory,
  36     Filesystem $filesystem
  37   }
  38   $this->testFactory = $testFactory;
  39   $this->transportBuilder = $transportBuilder;
  40   $this->inlineTranslation = $inlineTranslation;
  41   $this->storeManager = $storeManager;
  42   $this->fileUploaderFactory = $fileUploaderFactory;
  43   $this->filesystem = $filesystem;
  44   parent::__construct($context);
  45 }
  46
  47 public function execute()
  48 {
  49   $request_data = $this->getRequest()->getParams();
  50   $uploadedFile = $this->getRequest()->getFiles('attachment-pdf');
  51   //print_r($uploadedFile["name"]);die;
  52
  53   $data = [
  54     'name' => $request_data['name'],
  55     'email' => $request_data['email'],
  56     'phone' => $request_data['phone'],
  57     'location' => $request_data['location'],
  58     'attachment-pdf' => $uploadedFile["name"],
  59     'date_of_birth' => $request_data['date_of_birth'],
  60   ];
  61
  62   $testFactory = $this->testFactory->create();
  63   $test = $testFactory->setData($data);
  64   $test->save();
  65

```

Save.php - code - Visual Studio Code

```

  terminal  Help
  Save.php  ×
  Embitel > User > Controller > Index > Save.php
  63   $test = $testFactory->setData($data);
  64   $test->save();
  65
  66   if ($test->getUser_id()) {
  67     $this->messageManager->addSuccessMessage(__('Your Application has been sent.'));
  68
  69   // Send email to the customer
  70   $templateVars = [
  71     'data' => $data,
  72     'subject' => 'Job Application', // Set the subject
  73     'attachment_url' => '',
  74     'attachment_name' => '',
  75   ];
  76
  77   // $from = ['email' => 'shashankganesh53@gmail.com', 'name' => 'Shashank'];
  78   // $to = [$data['email'], $data['name']];
  79   $from = ['email' => $data['email'], 'name' => $data['name']];
  80   $to = ['shashankganesh53@gmail.com', 'Shashank'];
  81
  82
  83   $storeId = $this->storeManager->getStore()->getId();
  84
  85   try {
  86     $storeScope = \Magento\Store\Model\ScopeInterface::SCOPE_STORE;
  87     $templateOptions = [
  88       'area' => \Magento\Framework\App\Area::AREA_FRONTEND,
  89       'store' => $storeId,
  90     ];
  91
  92     $this->inlineTranslation->suspend();
  93     $transport = $this->transportBuilder
  94       ->setTemplateIdentifier('user confirmation')

```

terminal Help

Save.php - code - Visual Studio Code

Save.php x

```

Embitel > User > Controller > Index > Save.php

92     $this->inlineTranslation->suspend();
93     $transport = $this->transportBuilder
94         ->setTemplateIdentifier('user_confirmation')
95         ->setTemplateOptions($templateOptions)
96         ->setTemplateVars($templateVars)
97         ->setFrom($from)
98         ->addTo($to)
99         ->getTransport();

100    // Get the uploaded file information
101    if ($uploadedFile && !empty($uploadedFile['name'])) {
102        try {
103            $uploader = $this->fileUploaderFactory->create(['fileId' => 'attachment-pdf']);
104            $uploader->setAllowedExtensions(['pdf']);
105            // $uploader->setAllowRename(true);
106            $uploader->setFilesDispersion(true);
107            $destinationPath = $this->filesystem->getDirectoryRead(DirectoryList::MEDIA)->getAbsol
108            $result = $uploader->save($destinationPath);

109            if ($result['file']) {
110                $attachmentFile = $result['file'];
111                $attachmentFilePath = $destinationPath . $attachmentFile;

112                // Attach the file to the email
113                $attachmentPart = new Part(file_get_contents($attachmentFilePath));
114                $attachmentPart->setType(Mime::TYPE_OCTETSTREAM);
115                $attachmentPart->setDisposition(Mime::DISPOSITION_ATTACHMENT);
116                $attachmentPart->setEncoding(Mime::ENCODING_BASE64);
117                $attachmentPart->setFileName($uploadedFile['name']);

118                $body = $transport->getMessage()->getBody();
119
120                $body->addPart($attachmentPart);
121                $transport->getMessage()->setBody($body);

122                // Update the template variables
123                $templateVars['attachment_url'] = $this->storeManager->getStore()->getBaseUrl(\Mag
124                $templateVars['attachment_name'] = $uploadedFile['name'];
125            } else {
126                throw new LocalizedException(__('File upload failed.'));
127            }
128        } catch (\Exception $e) {
129            throw new LocalizedException(__('File upload failed: %1', $e->getMessage()));
130        }
131
132        $transport->sendMessage();
133        $this->inlineTranslation->resume();
134    } catch (\Exception $e) {
135        $this->messageManager->addErrorMassage(__('Error sending email.'));
136    }
137
138    if ($resultRedirect = $this->resultRedirectFactory->create()) {
139        $resultRedirect->setPath('user/index/index');
140        return $resultRedirect;
141    }
142
143    $this->messageManager->addErrorMassage(__('Data was not saved.'));
144
145
146
147
148
149
150
151
152

```

terminal Help

Save.php - code - Visual Studio Code

Save.php x

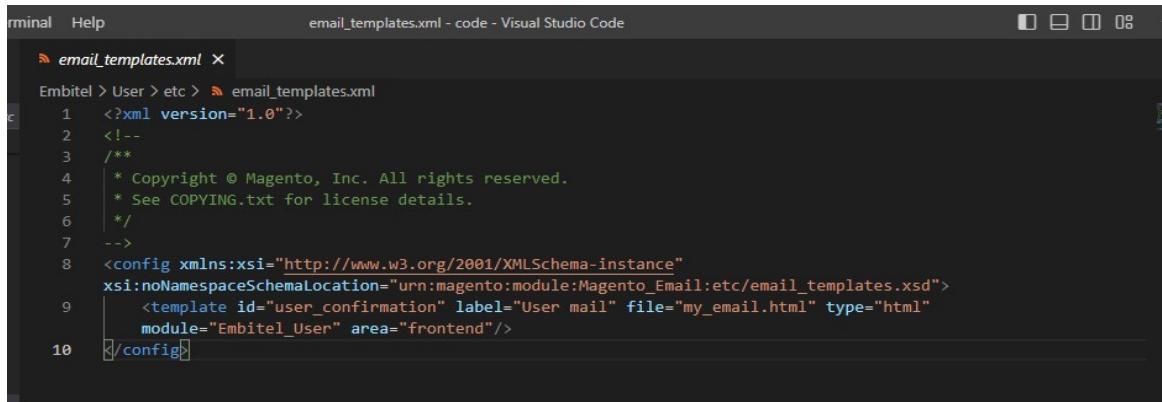
```

Embitel > User > Controller > Index > Save.php

121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152

```

1. For sending mail we need email_templates.xml which is placed inside etc



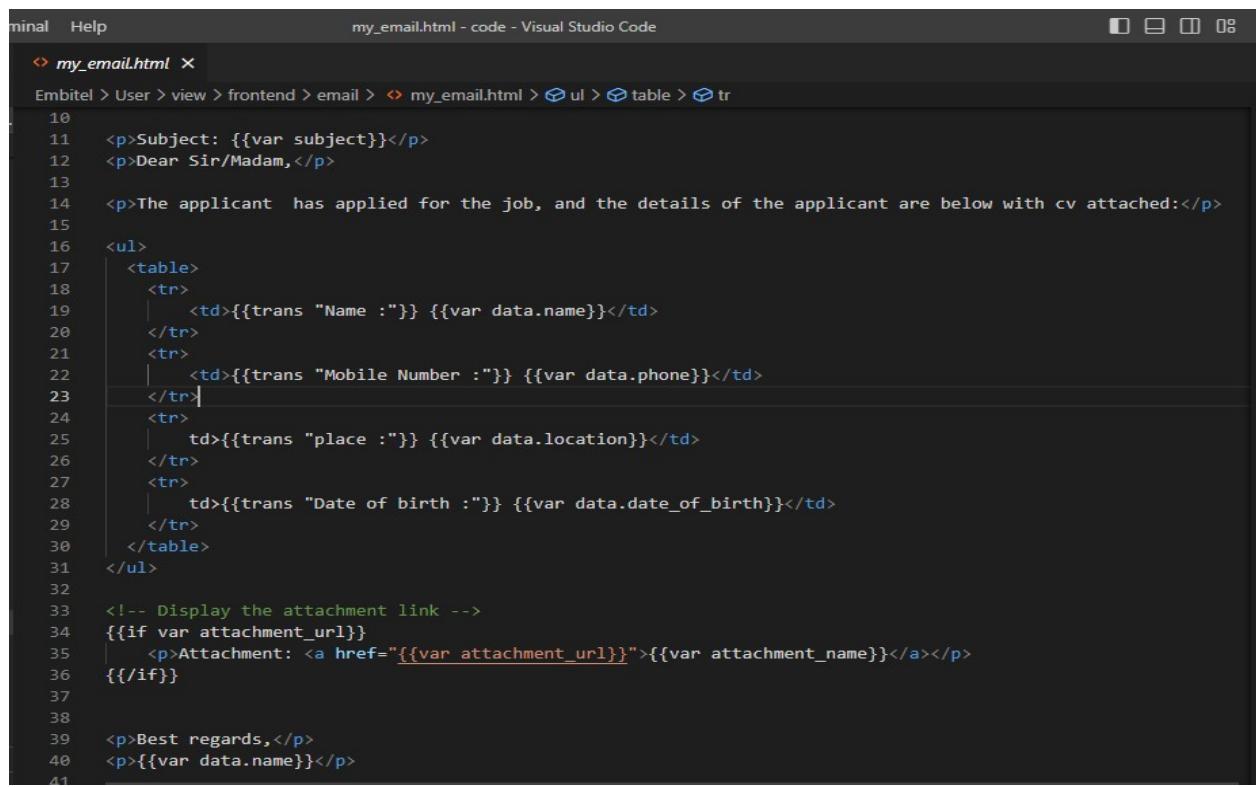
```

terminal Help
email_templates.xml - code - Visual Studio Code

email_templates.xml
Embitel > User > etc > email_templates.xml
1  <?xml version="1.0"?>
2  <!--
3  /**
4   * Copyright © Magento, Inc. All rights reserved.
5   * See COPYING.txt for license details.
6   */
-->
8 <config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:module:Magento_Email:etc/email_templates.xsd">
9   <template id="user_confirmation" label="User mail" file="my_email.html" type="html"
     module="Embitel_User" area="frontend"/>
10 </config>

```

- And to send a attachment and email content to a receiver we need to create a html file view/frontend/email/my_email.html



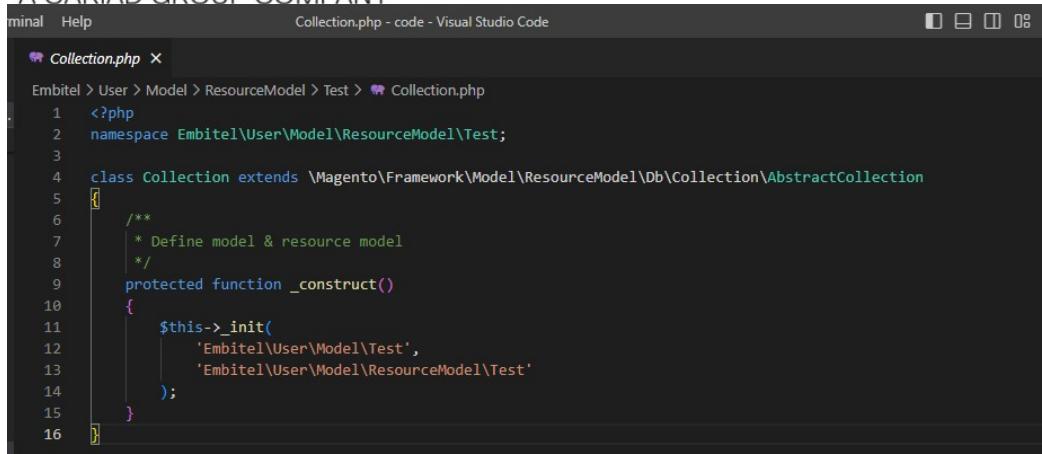
```

terminal Help
my_email.html - code - Visual Studio Code

my_email.html
Embitel > User > view > frontend > email > my_email.html > ul > table > tr
10
11  <p>Subject: {{var subject}}</p>
12  <p>Dear Sir/Madam,</p>
13
14  <p>The applicant has applied for the job, and the details of the applicant are below with cv attached:</p>
15
16  <ul>
17    <table>
18      <tr>
19        <td>{{trans "Name :"}} {{var data.name}}</td>
20      </tr>
21      <tr>
22        <td>{{trans "Mobile Number :"}} {{var data.phone}}</td>
23      </tr>
24      <tr>
25        <td>{{trans "place :"}} {{var data.location}}</td>
26      </tr>
27      <tr>
28        <td>{{trans "Date of birth :"}} {{var data.date_of_birth}}</td>
29      </tr>
30    </table>
31  </ul>
32
33  <!-- Display the attachment link -->
34  {{if var.attachment_url}}
35    <p>Attachment: <a href="{{var.attachment_url}}>{{var.attachment_name}}</a></p>
36  {{/if}}
37
38
39  <p>Best regards,</p>
40  <p>{{var data.name}}</p>
41

```

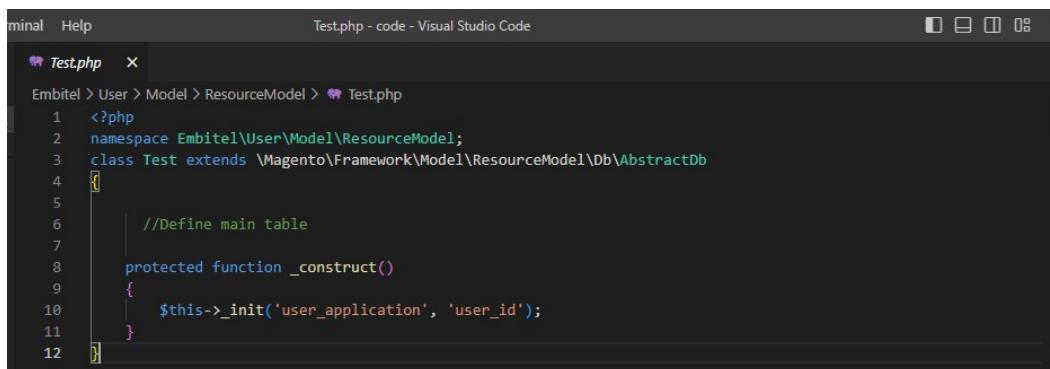
- And for saving the data in db table we need to create a model file and inside Resorcemodel/Test/Collection.php



Collection.php

```
Embitel > User > Model > ResourceModel > Test > Collection.php
1 <?php
2 namespace Embitel\User\Model\ResourceModel\Test;
3
4 class Collection extends \Magento\Framework\Model\ResourceModel\Db\Collection\AbstractCollection
5 {
6     /**
7      * Define model & resource model
8      */
9     protected function _construct()
10    {
11        $this->_init(
12            'Embitel\User\Model\Test',
13            'Embitel\User\Model\ResourceModel\Test'
14        );
15    }
16 }
```

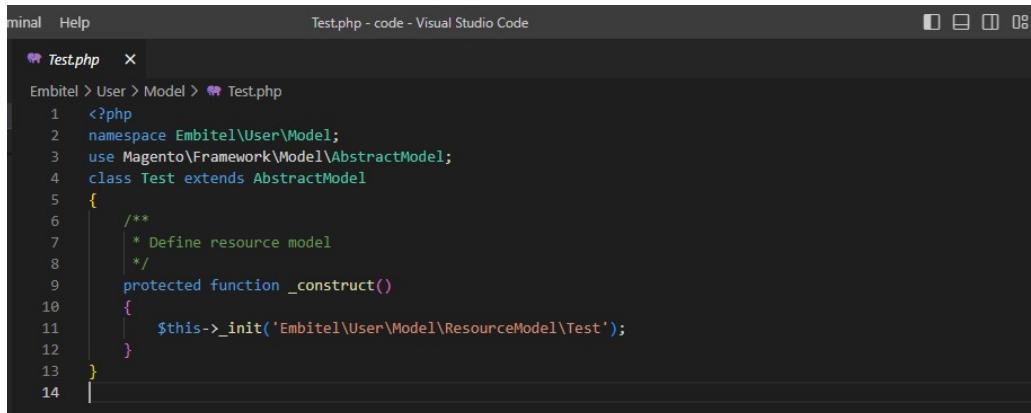
➤ Inside Resourcemodel test.php



Test.php

```
Embitel > User > Model > ResourceModel > Test.php
1 <?php
2 namespace Embitel\User\Model\ResourceModel;
3 class Test extends \Magento\Framework\Model\ResourceModel\Db\AbstractDb
4 {
5     /**
6      * Define main table
7      */
8     protected function _construct()
9     {
10        $this->_init('user_application', 'user_id');
11    }
12 }
```

➤ Inside Model test.php



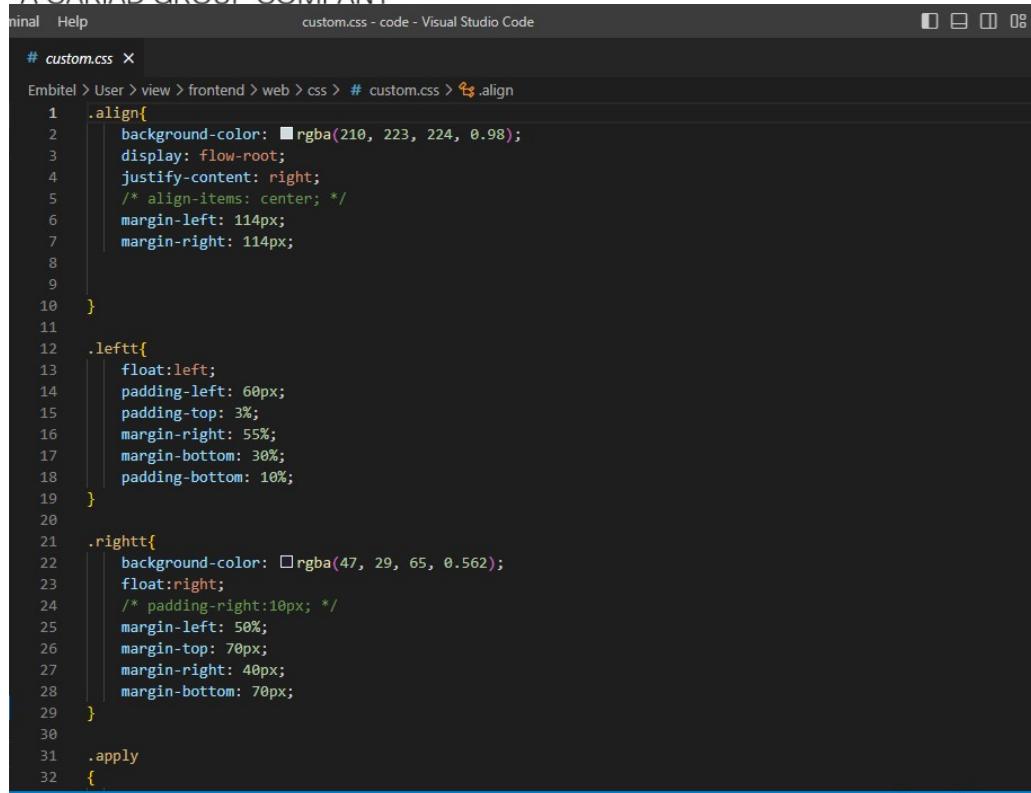
Test.php

```
Embitel > User > Model > Test.php
1 <?php
2 namespace Embitel\User\Model;
3 use Magento\Framework\Model\AbstractModel;
4 class Test extends AbstractModel
5 {
6     /**
7      * Define resource model
8      */
9     protected function _construct()
10    {
11        $this->_init('Embitel\User\Model\ResourceModel\Test');
12    }
13 }
```

➤ For adding css I created custom.css file the path for that is

➤ View\frontend\web\css\custom.css

➤



A screenshot of a Visual Studio Code window displaying a CSS file named "custom.css". The file contains several CSS rules for classes ".align", ".leftt", ".rightt", and ".apply". The ".align" rule sets a background color and applies justify-content: right; and margin-left: 114px; margin-right: 114px;. The ".leftt" rule uses float:left; and padding-left: 60px;. The ".rightt" rule uses float:right; and padding-right: 10px;. The ".apply" rule is partially visible at the bottom.

```
# custom.css x
Embitel > User > view > frontend > web > css > # custom.css > ⚙ .align
1  .align{
2    background-color: ■rgba(210, 223, 224, 0.98);
3    display: flow-root;
4    justify-content: right;
5    /* align-items: center; */
6    margin-left: 114px;
7    margin-right: 114px;
8
9
10 }
11
12 .leftt{
13   float:left;
14   padding-left: 60px;
15   padding-top: 3%;
16   margin-right: 55%;
17   margin-bottom: 30%;
18   padding-bottom: 10%;
19 }
20
21 .rightt{
22   background-color: □rgba(47, 29, 65, 0.562);
23   float:right;
24   /* padding-right:10px; */
25   margin-left: 50%;
26   margin-top: 70px;
27   margin-right: 40px;
28   margin-bottom: 70px;
29 }
30
31 .apply
32 {
```