Microsoft.AspNetCore.Mvc – provides classes and functionalities related to Asp.Net Core MVC
System.Diagnostics – classes for interacting with system processes, debugging and performance monitoring
System.IO – classes for input and output operations, such as reading from or writing to files or streams; handle file-related operations
System.Xml.Serialization – classes for serialization and deserialization
Serialization – converting objects to XML format Deserialization – converting XML data back into objects

XMLandSchemaFileUpload – model class used to bind form data on the view for uploading XML and schema files
IFormFile – an interface defined in Microsoft.AspNetCore.Mvc.Http namespace for the text stream received via HTTP from the network as an uploaded file.
XmlValidationError is the model class used to bind the list of errors found in the xml file to the view showing the validation result

```
public class XmlValidationError
{     [Display(Name = "XML Element")]         public string Element { get; set; }           }
```

```
using Lab3.Models;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using System.IO;
using System.Xml.Serialization;

namespace Lab3.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        [HttpGet]
        public IActionResult Index()
        {
            // deserialize data from xml
            // take local path
            string xmlFilePath = Path.GetFullPath("Data/restaurant_reviews.xml");
            // open file stream
            FileStream xmlFileStream = new FileStream(xmlFilePath, FileMode.Open);
            // serializer
            XmlSerializer xmlSerializerFile = new XmlSerializer(typeof(restaurants));
            // use serializer to deserialize data from file stream & turned to restaurants type
            restaurants restaurantsObject =
(restaurants)xmlSerializerFile.Deserialize(xmlFileStream);
            // close filestream
            xmlFileStream.Close();
            // return to frontend

            // creating the list to be returned
            List<RestaurantOverviewViewModel> restaurantList = new
List<RestaurantOverviewViewModel>();
            int i = 0;
            foreach(restaurantsRestaurant resto in restaurantsObject.restaurant)
            {
                // loop all restaurants and store as temp var resto
                RestaurantOverviewViewModel re = new RestaurantOverviewViewModel();
                re.Id = i++;
                re.Name = resto.generalInfo.name;
                re.FoodType = resto.generalInfo.cuisine;
                re.Rating = resto.generalInfo.rating.Value; // .Value because complex type
                re.Cost = resto.generalInfo.priceRange.Value;
                re.City = resto.generalInfo.address.city;
                re.ProvinceState = resto.generalInfo.address.state.ToString(); // .ToString() because
the state is an enumeration
                restaurantList.Add(re); // add everything to the list
            }

            return View(restaurantList);
        }

        // when a request is made to a URL mapped to a controller action method decorated with
[HttpGet], the method is invoked to handle the request
        // The action method can perform any necessary logic and return various types of responses,
not just views
        // View method is called to return a view to the clien
        [HttpGet] // decorated action method to be invoked when a GET request is made to the
corresponding URL
        public IActionResult Edit(int? id)
        {
            string xmlFilePath = Path.GetFullPath("Data/restaurant_reviews.xml");
            FileStream xmlFileStream = new FileStream(xmlFilePath, FileMode.Open);
            XmlSerializer xmlSerializerFile = new XmlSerializer(typeof(restaurants));
            restaurants restaurantsObject =
(restaurants)xmlSerializerFile.Deserialize(xmlFileStream);
            xmlFileStream.Close();

            RestaurantEditViewModel re = new RestaurantEditViewModel();
            // select specific restaurant
            restaurantsRestaurant specificResto = restaurantsObject.restaurant[(int) id]; // cast id
in int (can't be nullable)
            re.Id = (int) id;
            re.Name = specificResto.generalInfo.name;
            re.StreetAddress = specificResto.generalInfo.address.street;
            re.City = specificResto.generalInfo.address.city;
            re.ProvinceState = specificResto.generalInfo.address.state;
            re.PostalZipCode = specificResto.generalInfo.address.postalCode;
            re.Summary = specificResto.review.summary;
            re.Rating = specificResto.generalInfo.rating.Value;




            return View(re);
        }

        // allows you to handle form submissions, API requests, and other POST-based actions in your
application
        [HttpPost] // decorated action method that specifies that the action method should respond to
HTTP POST requests
        public IActionResult Edit(RestaurantEditViewModel model)
        {
            string xmlFilePath = Path.GetFullPath("Data/restaurant_reviews.xml");
            FileStream xmlFileStream = new FileStream(xmlFilePath, FileMode.Open); // reading
            XmlSerializer xmlSerializerFile = new XmlSerializer(typeof(restaurants));
            restaurants restaurantsObject =
(restaurants)xmlSerializerFile.Deserialize(xmlFileStream);
            xmlFileStream.Close();

            restaurantsRestaurant specificResto = restaurantsObject.restaurant[model.Id]; // taking
in the model & no need to cast
            specificResto.generalInfo.name = model.Name;
            specificResto.generalInfo.address.street = model.StreetAddress;
            specificResto.generalInfo.address.city = model.City;
            specificResto.generalInfo.address.state = model.ProvinceState;
            specificResto.generalInfo.address.postalCode = model.PostalZipCode;
            specificResto.review.summary = model.Summary;
            specificResto.generalInfo.rating.Value = (byte) model.Rating;

            // serializing/saving/writing
            xmlFileStream = new FileStream(xmlFilePath, FileMode.Create); // reopening & writing
            xmlSerializerFile.Serialize(xmlFileStream, restaurantsObject); // one tied to the id
            xmlFileStream.Close();

            return RedirectToAction("Index");
        }
```

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified"
           elementFormDefault="qualified"
           targetNamespace="http://www.algonquincollege.com/cst8259/labs"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:alg="http://www.algonquincollege.com/cst8259/labs">

    <!-- global element: restaurants -->
    <xs:element name="restaurants">
        <xs:complexType>
            <xs:sequence>

                <!-- restaurant -->
                <xs:element maxOccurs="unbounded" name="restaurant">
                    <xs:complexType>
                        <xs:sequence>


    <!-- global element: address -->
    <xs:element name="address">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="street" type="xs:string"/>
                <xs:element name="city" type="xs:string"/>
                <xs:element name="state" type="alg:state"/>
                <xs:element name="postalCode" type="alg:postalCode"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <!-- global element: state  (simple type of string with restriction of CA
provinces standard code)-->
    <xs:simpleType name="state">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ON"/>
            <xs:enumeration value="QC"/>
            <xs:enumeration value="BC"/>
            <xs:enumeration value="AB"/>
            <xs:enumeration value="SK"/>
            <xs:enumeration value="MB"/>
            <xs:enumeration value="NL"/>
            <xs:enumeration value="PE"/>
            <xs:enumeration value="NB"/>
            <xs:enumeration value="NS"/>
            <xs:enumeration value="NT"/>
            <xs:enumeration value="NV"/>
            <xs:enumeration value="YK"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- global element: postalCode (simple type of string with regex restriction)
-->
    <xs:simpleType name="postalCode">
        <xs:restriction base="xs:string">
            <xs:pattern value="^[a-zA-Z]\d[a-zA-Z][ -]?\d[a-zA-Z]\d$"/>
        </xs:restriction>
    </xs:simpleType>

    <!-- global element: menuItem -->
    <xs:element name="menuItem">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="name" type="xs:string" />
                <xs:element name="description" type="xs:string" />
                    <xs:element name="price" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:simpleContent>
                                <xs:extension base="xs:decimal">
                                    <xs:attribute name="quantity"
type="xs:unsignedInt" />
                                </xs:extension>
                            </xs:simpleContent>
                        </xs:complexType>
                </xs:element>
        </xs:sequence>
        <xs:attribute name="type" type="xs:string" />
        </xs:complexType>
    </xs:element>

</xs:schema>
```

```
 [HttpGet] // asking for data
public IActionResult Index()
       { return View(); }

       [HttpPost] // sending data
       public IActionResult Index(XMLandSchemaFileUpload dataFile)
       {
           // set up files
           IFormFile xmlFile = dataFile.XmlFile; // from XMLandSchemaFileUpload.cs
           IFormFile xsdFile = dataFile.SchemaFile;

           if (ModelState.IsValid)
           {
               // connect schema file with backend
               // schema -converting
               XmlReader schemaReader = XmlReader.Create(xsdFile.OpenReadStream());
               XmlSchemaSet schemaSet = new XmlSchemaSet();
               // add reader to set
               schemaSet.Add(null, schemaReader);

               // set up setting for schema
               XmlReaderSettings settings = new XmlReaderSettings();
               // set up settings
               settings.ValidationType = ValidationType.Schema;
               settings.Schemas = schemaSet;
               // schema is in the backend by then

               // event handler - create the errors
               List<XmlValidationError> errors = new List<XmlValidationError>();
               settings.ValidationEventHandler +=
                   (s, e) => errors.Add(new XmlValidationError // event handler for errors
                   {
                       Element = ((XmlReader)s).Name, // need to cast converts s to XmlReader type -
auto convert to string
                       ErrorType = e.Severity.ToString(), // e.Severity
                       Line = e.Exception.LineNumber,
                       Column = e.Exception.LinePosition,
                       Message = e.Message
                   });

               // xml
               XmlReader xmlReader = XmlReader.Create(xmlFile.OpenReadStream(), settings);
               // importing xml and validating using the schema
               while (xmlReader.Read()) { } // trigger the event handler for error
               ViewBag.XmlFileName = xmlFile.FileName;
               ViewBag.XsdFileName = xsdFile.FileName;
               return View("ValidationResult", errors);
           }
           return View();
       }
```

DOM (Document object model) – uses a set of standardized objects to represent structured document (data) while they are held in memory
- Create XML documents, Navigate an XML document, Add/modify/delete parts of XML documents

XmlTextWriter is a class that enables you to output XML to files or an open stream (such as a network socket).

```
XmlSerializer serializor= new XmlSerializer(typeof(BookOrder))
```

XML Schema - is a file used to define what elements and attributes may appear in an XML document; also defines the relationship of the elements and what data may be stored in them

The SimpleXMLElement class is the central class for all operations within this extension.

```
$xml = "<root><node1>content</node1></root>";
$sxe = new SimpleXMLElement($xml);
```

Or load and XML from a file:

```
$sxe = simplexml_load_file("filename.xml");
```

Repeating child elements can also be accessed using 0 based index.

```
print "<h1>".$book->chapter->title."</h1>"
```

When object is accessed without index, the first element in the array is accessed

```
print "<p>".$para."</p>";
print "<p>".$para[1]."</p>";
```

```php
<?php
session_start();
include("./Common/Header.php");

// Restaurant dropdown
$restaurantArray = [];
$xmlFile = simplexml_load_file("restaurant_reviews.xml");
foreach ($xmlFile as $restaurant) {
    array_push($restaurantArray, $restaurant);
}

// Restaurant selection
$isRestaurantSelected = false;
$selectedRestaurant = "-1";
// if set
if (isset($_POST["restaurantsDropdown"]) && $_POST["restaurantsDropdown"] != "-1") {
    $isRestaurantSelected = true;
    $selectedRestaurant = $_POST["restaurantsDropdown"];

    $specificRestaurant = [];
    foreach ($restaurantArray as $currentRestaurant) {
        if ($currentRestaurant->generalInfo->name == $selectedRestaurant) {
            $specificRestaurant = $currentRestaurant;
        }
    }

    //var_dump($specificRestaurant);

    // assigning variables for all elements
    // variables for all info
    $streetAddress = $specificRestaurant->generalInfo->address->street;
    $city = $specificRestaurant->generalInfo->address->city;
    $province = $specificRestaurant->generalInfo->address->state;
    $postalCode = $specificRestaurant->generalInfo->address->postalCode;
    $summary = $specificRestaurant->review->summary;
    $rating = $specificRestaurant->generalInfo->rating;

    $ratingMin = $specificRestaurant->generalInfo->rating["min"];
    $ratingMax = $specificRestaurant->generalInfo->rating["max"];

    //var_dump($rating);
    //var_dump($ratingMin);
    //var_dump($ratingMax);
}

// success notification and saving changes
$success = false;
if (isset($_POST["saveButton"])) {
    $selectedRestaurant = $_POST["restaurantsDropdown"];

    foreach ($xmlFile as $currentRestaurant) {
        if ($currentRestaurant->generalInfo->name == $selectedRestaurant) {
            $currentRestaurant->generalInfo->address->street = $_POST["streetAddressInput"];
            $currentRestaurant->generalInfo->address->city = $_POST["cityInput"];
            $currentRestaurant->generalInfo->address->state = $_POST["provinceInput"];
            $currentRestaurant->generalInfo->address->postalCode = $_POST["postalCodeInput"];
            $currentRestaurant->review->summary = $_POST["summaryInput"];
            $currentRestaurant->generalInfo->rating = $_POST["ratingChoice"];
        }

        //var_dump($_POST["saveButton"]);
    }
    $xmlFile->asXML("restaurant_reviews.xml");
    $success = true;
}

//var_dump($isRestaurantSelected);
//var_dump($_POST["restaurantsDropdown"]);
?>

<!-- HTML STUFF -->
<div class="container">

    <!-- Title -->
    <div class="row">
        <div class="col-md-6 col-md-offset-2">
            <h1>Online Restaurant Review</h1>
        </div>
    </div>
    <p>Select a restaurant from the drop down list to view/edit its review.</p>

    <!-- Form -->
    <form action="index.php" method="post">


        <!-- Select Restaurant Drop down -->
        <div class="row form-group">

            <div class="col-md-2">
                <label>Restaurant:</label>
            </div>
            <div class="col-md-8">
                <select name="restaurantsDropdown" class="form-control" onchange="selectionEvent()">
                    <option value="-1">Select Restaurant...</option>
                    <?php
                    foreach ($restaurantArray as $restaurant) {
                        ?>
                        <option
                        <?php
                        if ($selectedRestaurant == $restaurant->generalInfo->name) {
                            echo "selected";
                        }
                        ?>
                            value="<?php echo $restaurant->generalInfo->name ?>">
                                <?php echo $restaurant->generalInfo->name ?>
                        </option>
                        <?php
                    }
                    ?>
                </select>
            </div>
        </div>

        <?php
        if ($isRestaurantSelected) {
            ?>

            <!-- Street Address -->
            <div class="row form-group">
                <div class="col-md-2">
                    <label>Street Address: </label>
                </div>
                <div class="col-md-8">
                    <input type="text" name="streetAddressInput" class="form-control"
value="<?php echo $streetAddress ?>"></input>
                </div>
            </div>


            <!-- City -->
            <div class="row form-group">
                <div class="col-md-2">
                    <label>City: </label>
                </div>
                <div class="col-md-8">
                    <input type="text" name="cityInput" class="form-control" value="<?php
echo $city ?>"></input>

                </div>
            </div>

            <!-- Province/State -->
            <div class="row form-group">
                <div class="col-md-2">
                    <label>Province/State:</label>
                </div>
                <div class="col-md-8">
                    <input type="text" name="provinceInput" class="form-control"
value="<?php echo $province ?>"></input>
                </div>
            </div>

            <!-- Postal/Zip Code -->
            <div class="row form-group">
                <div class="col-md-2">
                    <label>Postal Code: </label>
                </div>
                <div class="col-md-8">
                    <input type="text" name="postalCodeInput" class="form-control"
value="<?php echo $postalCode ?>"> </input>
                </div>
            </div>

            <!-- Summary -->
            <div class="row form-group">
                <div class="col-md-2">
                    <label>Summary: </label>
                </div>
                <div class="col-md-8">
                    <textarea type="text" rows="8" class="form-control"
name="summaryInput"><?php echo $summary ?></textarea>
                </div>
            </div>

            <!-- Rating -->
            <div class="row form-group">
                <div class="col-md-2">
                    <label>Rating: </label>
                </div>
                <div class="col-md-8">
                    <select name="ratingChoice" class="form-control">
                        <?php
                        for ($i = 1; $i <= $ratingMax; $i++) {
                            ?>
                            <option
                            <?php
                            if ($i == $rating) {
                                echo "selected";
                            }
                            ?>
                                value="<?php echo $i ?>"><?php echo $i ?>
                            </option>
                            <?php
                        }
                        ?>
                    </select>
                </div>
            </div>

            <!-- Save Button -->
            <div class="row  form-group">
                <div class="col-md-10 col-md-offset-2">
                    <input type='submit' class="btn btn-primary btn-min-width"
name='saveButton' value="Save Changes" />
                </div>
            </div>

        <?php }
        ?>
        <!-- Confirmation -->
        <?php if ($success) { ?>
            <div class="row">
                <div class="col-md-10">
                    <label id="successNotification" class="form-control alert-success">
                        Revised Restaurant Review has been saved to:
restaurant_reviews.xml
                    </label>
                </div>
            </div>
        <br/>
        <?php } ?>
        <input hidden type="submit" id="updateButton" />

    </form>

</div>

<?php
include_once('./Common/Footer.php');
?>
```

```php
<?php
    $book = simplexml_load_file('Book.xml');
    $book->chapter->para[1] = ""; //empty an element
    unset($book->chapter->title); //remove an element
?>
```