- **What is Nuget & package?**

  NuGet: NuGet is a package manager for .NET. It allows developers to create, share, and consume useful .NET libraries (packages) easily.

  Package: A package is a compiled library or tool that can be added to a project to provide additional functionality without the need to write that code from scratch.

- **What is WebService?**

  A WebService is a standardized way of integrating web-based applications using the XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone.

- **What is request and response?**

  Request: When a client (browser or application) sends data to the server, asking for some resources or information.

  Response: The data sent back from the server to the client in answer to the request.

- **What is postback?**

  In ASP.NET, a postback is the process of submitting an ASP.NET page to the server for processing. A postback is usually triggered by events such as button clicks.

- **What is IIS?**

  IIS is a flexible, secure, and manageable Web server for hosting anything on the Web. From media streaming to web applications, IIS is a server role in Windows Server.

- **What is web.config?**

  web.config is a configuration file for ASP.NET applications. It is an XML file that stores configuration settings for the application such as database connections, session states, error handling, etc.

- **Which is the type of WebService?**

  SOAP (Simple Object Access Protocol) Web Services

  REST (Representational State Transfer) Web Services

- **What is XML and Json?**

  XML (Extensible Markup Language): A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

  JSON (JavaScript Object Notation): A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.

- **Create application to perform crud operation using WebApis.**

&&

- **Create Web Grid Example using Nuget Package for City table data Display with country name and state name using WebApi.**

Model: C#

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
}

public class Country
{
    public int CountryId { get; set; }
    public string CountryName { get; set; }
    public ICollection<State> States { get; set; }
}

public class State
{
    public int StateId { get; set; }
    public string StateName { get; set; }
    public int CountryId { get; set; }
    public Country Country { get; set; }
    public ICollection<City> Cities { get; set; }
}

public class City
{
    public int CityId { get; set; }
    public string CityName { get; set; }
    public int StateId { get; set; }
    public State State { get; set; }
}
```

Model – 2: C#

```
using System.Data.Entity;

public class ApplicationDbContext : DbContext
{
   public DbSet<Product> Products { get; set; }
   public DbSet<City> Cities { get; set; }
    public DbSet<State> States { get; set; }
    public DbSet<Country> Countries { get; set; }
}
```

Model View: C#

```
public class CityViewModel
{
   public int CityId { get; set; }
   public string CityName { get; set; }
   public string StateName { get; set; }
   public string CountryName { get; set; }
}
```

WebApi Controller: C#

```
using System.Linq;
using System.Net;
using System.Web.Http;
using System.Web.Http.Description;
using MobileApp.Models;
```

```
public class ProductsController : ApiController
{
   private ApplicationDbContext db = new ApplicationDbContext();

   // GET: api/Products
   public IQueryable<Product> GetProducts()
   {
      return db.Products;
   }

   // GET: api/Products/5
   [ResponseType(typeof(Product))]
   public IHttpActionResult GetProduct(int id)
   {
      Product product = db.Products.Find(id);
      if (product == null)
      {
         return NotFound();
      }

      return Ok(product);
   }

   // PUT: api/Products/5
   [ResponseType(typeof(void))]
   public IHttpActionResult PutProduct(int id, Product product)
   {
      if (!ModelState.IsValid)
      {
         return BadRequest(ModelState);
      }

      if (id != product.Id)
      {
         return BadRequest();
      }

      db.Entry(product).State = EntityState.Modified;

      try
      {
         db.SaveChanges();
      }
      catch (DbUpdateConcurrencyException)
      {
         if (!ProductExists(id))
         {
             return NotFound();
         }
         else
         {
             throw;
         }
      }

      return StatusCode(HttpStatusCode.NoContent);
   }

   // POST: api/Products
```

```
[ResponseType(typeof(Product))]
public IHttpActionResult PostProduct(Product product)
{
   if (!ModelState.IsValid)
   {
      return BadRequest(ModelState);
   }

   db.Products.Add(product);
   db.SaveChanges();

   return CreatedAtRoute("DefaultApi", new { id = product.Id }, product);
}

// DELETE: api/Products/5
[ResponseType(typeof(Product))]
public IHttpActionResult DeleteProduct(int id)
{
   Product product = db.Products.Find(id);
   if (product == null)
   {
      return NotFound();
   }

   db.Products.Remove(product);
   db.SaveChanges();

   return Ok(product);
}
```

```
    protected override void Dispose(bool disposing)
    {
        if (disposing)
        {
            db.Dispose();
        }
        base.Dispose(disposing);
    }

    private bool ProductExists(int id)
    {
        return db.Products.Count(e => e.Id == id) > 0;
    }
}
```

WebApi Controller – 2: C#

```
using System.Linq;
using System.Net;
using System.Web.Http;
using System.Web.Http.Description;
using MobileApp.Models;

public class CitiesController : ApiController
{
    private ApplicationDbContext db = new ApplicationDbContext();

    // GET: api/Cities
    public IQueryable<object> GetCities()
    {
        var cities = db.Cities.Include("State.Country").Select(c => new
        {
            c.CityId,
            c.CityName,
            StateName = c.State.StateName,
            CountryName = c.State.Country.CountryName
        });

        return cities;
    }

    // Other CRUD methods here...

    protected override void Dispose(bool disposing)
    {
        if (disposing)
        {
            db.Dispose();
        }
        base.Dispose(disposing);
    }
}
```

Controller : C#

```
public class HomeController : Controller
{
    private static readonly HttpClient client = new HttpClient();

    public async Task<ActionResult> Index()
    {
        var response = await
client.GetAsync("http://localhost:your_port/api/Products");
        var products = await
response.Content.ReadAsAsync<IEnumerable<Product>>();
        return View(products);
    }

    public ActionResult Create()
    {
        return View();
    }

    [HttpPost]
    public async Task<ActionResult> Create(Product product)
    {
        if (ModelState.IsValid)
        {
            await
client.PostAsJsonAsync("http://localhost:your_port/api/Products", product);
            return RedirectToAction("Index");
        }
        return View(product);
    }

    public async Task<ActionResult> Edit(int id)
    {
        var response = await
client.GetAsync($"http://localhost:your_port/api/Products/{id}");
        var product = await response.Content.ReadAsAsync<Product>();
        return View(product);
    }
    [HttpPost]
    public async Task<ActionResult> Edit(Product product)
    {
        if (ModelState.IsValid)
        {
            await
client.PutAsJsonAsync($"http://localhost:your_port/api/Products/{product.Id}
", product);
            return RedirectToAction("Index");
        }
        return View(product);
    }

    public async Task<ActionResult> Delete(int id)
    {
        var response = await
client.GetAsync($"http://localhost:your_port/api/Products/{id}");
        var product = await response.Content.ReadAsAsync<Product>();
        return View(product);
    }

    [HttpPost, ActionName("Delete")]
    public async Task<ActionResult> DeleteConfirmed(int id)
    {
        await
client.DeleteAsync($"http://localhost:your_port/api/Products/{id}");
        return RedirectToAction("Index");
    }
}
```

Controller – 2: C#

```
public class HomeController : Controller
{
    private static readonly HttpClient client = new HttpClient();

    public async Task<ActionResult> Index()
    {
        var response = await
client.GetAsync("http://localhost:your_port/api/Cities");
        var cities = await
response.Content.ReadAsAsync<IEnumerable<CityViewModel>>();
        return View(cities);
    }
}
```

Grid View: Html

```
@model IEnumerable<MobileApp.Models.CityViewModel>
@{
    ViewBag.Title = "Cities";
}

<h2>Cities</h2>

@grid.GetHtml(
    tableStyle: "table table-striped",
    headerStyle: "thead-dark",
    columns: grid.Columns(
        grid.Column("CityId", "City ID"),
        grid.Column("CityName", "City Name"),
        grid.Column("StateName", "State Name"),
        grid.Column("CountryName", "Country Name")
    )
)
```

Index View: Html

```
@model IEnumerable<MobileApp.Models.Product>
@{
    ViewBag.Title = "Product List";
}

<h2>Product List</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>

<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Price)
        </th>
        <th></th>
    </tr>
    @foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Price)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.Id })
            </td>
        </tr>
    }
</table>
```

Create View: Html

```
@model MobileApp.Models.Product
@{
    ViewBag.Title = "Create Product";
}

<h2>Create Product</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
```

```
        <h4>Product</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Price, htmlAttributes: new { @class
= "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Price, new { htmlAttributes =
new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Price, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}
```

```
<div>
   @Html.ActionLink("Back to List", "Index")
</div>
```

Edit View: Html

```
@model MobileApp.Models.Product
@{
   ViewBag.Title = "Edit Product";
}

<h2>Edit Product</h2>

@using (Html.BeginForm())
{
   @Html.AntiForgeryToken()

   @Html.HiddenFor(model => model.Id)

   <div class="form-horizontal">
      <h4>Product</h4>
      <hr />
      @Html.ValidationSummary(true, "", new { @class = "text-danger" })
      <div class="form-group">
         @Html.LabelFor(model => model.Name, htmlAttributes: new {
@class = "control-label col-md-2" })
         <div class="col-md-10">
            @Html.EditorFor(model => model.Name, new { htmlAttributes =
new { @class = "form-control" } })
```

```
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Price, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Price, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Price, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

Delete View: Html

```
@model MobileApp.Models.Product
@{
    ViewBag.Title = "Delete Product";
}

<h2>Delete Product</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Product</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Name)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Name)
        </dd>
        <dt>
            @Html.DisplayNameFor(model => model.Price)
        </dt>
        <dd>
            @Html.DisplayFor(model => model.Price)
        </dd>
    </dl>

    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()
        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```