**Con Edison - Retail Choice v 4.2.6**

**Index**

1. Requirement Overview ……………………………………………………… 2
2. Architecture ……………………………………………………… 3
3. Functional Overview ……………………………………………………… 4
4. Retail Choice Web Debugging ……………………………………………………… 5
5. Retail Choice Web Deployment ……………………………………………………… 7
6. Code Structure ……………………………………………………… 9
7. Application Screen Shots ……………………………………………………… 36
8. **Requirement Overview**

Retail Choice has been one of the primary applications of Con Edison which is an interface for all the other small gas and electricity companies and their customers. All customers can view their account information, download various kinds of reports, view invoice and bill history, access meter readings history, usage forecast report, interval data and more.

Current version of the application is using old technologies with classic asp, new version is expected to use latest Microsoft technologies with high end client server application techniques. Along with upgrading the technologies the new AMI Account Meter Interval module will be integrated to the application.

AMI is the new Account Meter Interval data. The so called Smart Meter will be installed to the customers who are requesting for it. These smart meters automatically sent the meter reading data to the Con Edison servers at regular intervals of time in a day avoiding the manual meter reading. With this data it would be easy to analyze and understand the usage and also to manage the forecast of the usage especially for big customer’s like banks, shopping malls, companies, tall raise apartments and more. As the size of this data is huge additional software updates are required to handle this huge volume of data and to process them and use them for analysis. Customer can make a request of monthly and historical usage data of these meter intervals.

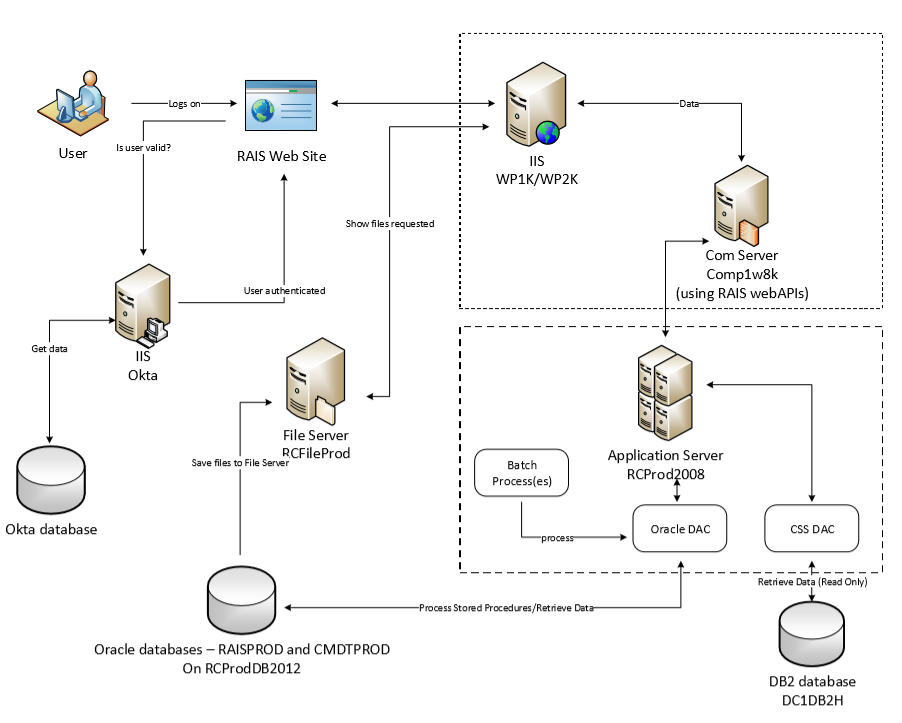
Transaction tracking system is also another small module which has to be integrated. With this customers can track the following transactions like Account Update, New Request, Price Change, Drop Request and more. This module adds more transparency to the system workflow and customer

Account Credit and Debit module is another important update to the system, this is basically Con Edison and its marketers giving credit and debit back to the customers. This also involves the tracking system to track the correct previous transactions for credit and debit.

Centralized file storage system which is on physical drive and this file to be transferred to the customer to download using modern encryption techniques. Single sign on Okta server will be integrated for user authentication process.

1. **Architecture**

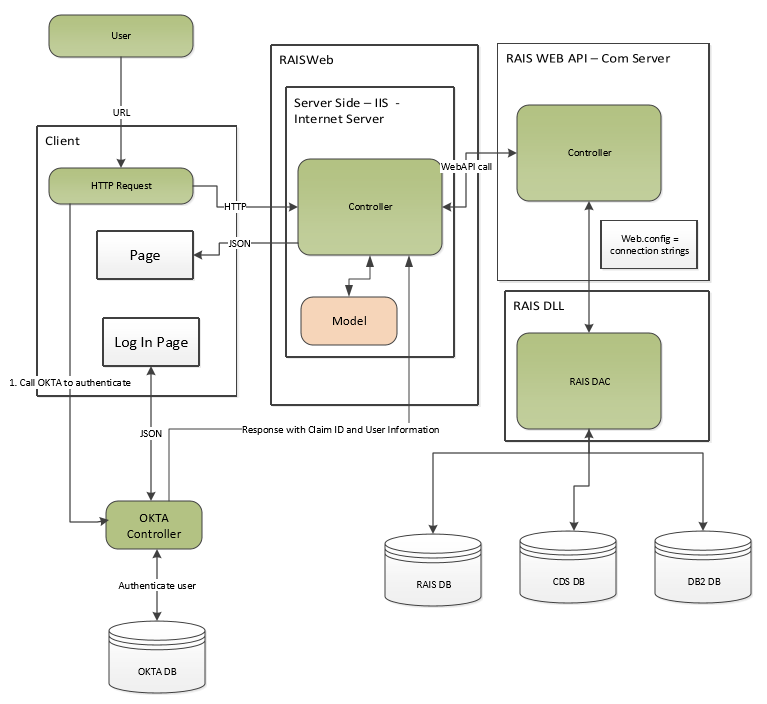
Architecture of the application is as shown in the following diagram wherein we can see two application servers WP1k, WP2K placed at data center, a file server, single sign on Okta server for user authentication process, internal com sever and the database servers.

****

1. **Functional Overview**

Functional overview is as shown below where we can see the end to end request workflow.

1. Client makes a request from any of the browser with the URL of Retail Choice.
2. Request reaches the app web server where IIS is installed and hosts Retail Choice Application
3. Based on the type of the request, app server makes a web api call to the com server which is present inside the Con Edison environment.
4. Web Api then connects to the core internal database of Con Edison mainly MS SQL server, Oracle server and IBM DB2 server and get the required data for the user.
5. This response is sent back to the web api and then to the app server and then to the client.



1. **Retail Choice Web Debugging**

Solution mainly has 2 web applications projects.

1. RAISWeb
2. RAISWebApi

Application flow is here

**RAISWeb project🡪 RAISWebAPI project 🡪 Businees project 🡪 Dac project**

RAISWeb

* RAISWeb is an asp.net MVC external facing application.
* To make Okta login work locally and test, RAISWeb project has to be run on IIS express using the port number **55312** like this <http://localhost:55312/>, this port number is registered to RAISWeb application in Okta settings so we cannot change the port number at any time until we change this in Okta.
* To debug and test Okta we need to have breakpoint and look at startup.cs file and claimextension.cs file if we have any issues.
* To debug RAISWeb we need to start the project in Visual Studio
* Each time RAISWeb makes a call to RAISWebApi to get the data.

RAISWebApi

* To debug RAISWebApi, we need to attach to process of w3wp.exe after start the RAISWeb project from visual studio.
* If you don’t find w3wp.exe then go to internet explorer and browse the webapi, for example <http://localhost/RAISWebApi/api/Alerts/>

**Logs**

Log file is created at following locations in their corresponding environment.

Local, Dev, Test - [\\rcfiletest\RetailChoiceArchives\RAIS\LogFiles\RAISWeb](file:///\\rcfiletest\RetailChoiceArchives\RAIS\LogFiles\RAISWeb)

Prod - [\\rcfileprod\RetailChoiceArchives\RAIS\LogFiles\RAISWeb](file:///\\rcfileprod\RetailChoiceArchives\RAIS\LogFiles\RAISWeb)

Each day a new log file is created and we need to manually delete old log files once a month or two.

Each time where there is an exception an email is sent and it is logged into the file.

RAISWeb which is an external facing application does not have rights to write to a file on file server, so call the WebApi to do this.

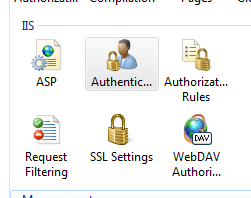
RAISWebApi have permission to write directly to log file as we are running the app pool under specific identity.

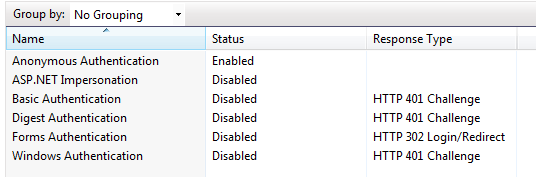
1. **Retail Choice Web Deployment**

* There are 2 web applications RetailChoice and RAISWebApi, one is ASP.NET MVC internet application and another is web API which is an intranet web service application both developed in MVC 5.0.
* Internet application (**RetailChoice**) is hosted on internet server and web API (**RAISWebApi**) is hosted on Com server.

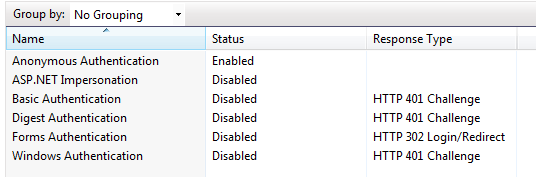
Steps to deploy **RetailChoice** internet application on internet application

1. Create a new app pool **RAISWeb** on internet server
2. Create a new virtual directory on internet server with name RetailChoice and map it to RAISWebsite folder inside RaisWebsiteSln. Assign the above created app pool to this virtual directory.
3. In IIS go to Authentication in contents view for RetailChoice virtual directory, enable only Anonymous Authentication and disable rest of them.





Steps to deploy RAISWebApi intranet application on COM server

1. Create a new app pool **RAISWeb** on intranet COM server.
2. Assign the identity for the app pool to windows accounts **RC\_INTERNET\_TEST** in development and test and **RC\_INTERNET\_PROD** in production environment.
3. In Dev only set “Enable 32 bit” to true on the RAISWeb app pool in com server hosting RAISWebApi application, this will make oracle work.
4. Make sure we have connections asp file exists for RAISWebAPi on COM server
5. Create a new virtual directory on intranet COM server with name **RAISWebApi** and map it to RAISWebApi folder inside RaisWebsiteSln. Assign the above created app pool RAISWeb to this virtual directory.
6. In IIS go to Authentication in contents view for RAISWebAPi virtual directory, enable only Anonymous Authentication and disable rest of them. 
7. **Code Structure**

**Controllers**

**Account Controller – handles user related actions**

using System;

using System.Web.Mvc;

using System.Web;

using Microsoft.Owin.Security;

using Microsoft.AspNet.Identity;

using System.Linq;

using ConEdison.RAISWeb.Common;

using ConEdison.RAISWeb.EntityTypes;

using ConEdison.RAISWeb.Interfaces;

namespace RAISWeb.Controllers

{

[Authorize]

public class AccountController : BaseController

{

public ActionResult Login(string returnUrl)

{

RAISUser raisUser = null;

bool IsBoth = false;

string[] strGroups = null;

string[] strGroupDetails = null;

return View();

if (Session["User"] == null)

{

if (User != null)

{

raisUser = new RAISUser();

try

{

if (User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().Contains("conedinternal") &&

User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().Contains("onrinternal"))

{

IsBoth = true;

raisUser.Role = UserRole.InternalAdmin;

LocalUtils.WriteLog(User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName"));

}

else if (User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().Contains("conedinternal"))

{

raisUser.Company = ConEdison.RAISWeb.Interfaces.UserCompany.Coned;

raisUser.Role = UserRole.InternalAdmin;

LocalUtils.WriteLog(User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName"));

}

else if (User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().Contains("onrinternal"))

{

raisUser.Company = ConEdison.RAISWeb.Interfaces.UserCompany.Oru;

raisUser.Role = UserRole.InternalAdmin;

LocalUtils.WriteLog(User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName"));

}

else if (User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().StartsWith("rcc"))

{

raisUser.Company = ConEdison.RAISWeb.Interfaces.UserCompany.Coned;

LocalUtils.WriteLog(User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName"));

}

else if (User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().StartsWith("rco"))

{

raisUser.Company = ConEdison.RAISWeb.Interfaces.UserCompany.Oru;

LocalUtils.WriteLog(User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName"));

}

if (raisUser.Role == null)

{

LocalUtils.WriteLog(User.GetClaimValue("http://schemas.microsoft.com/ws/2008/06/identity/claims/role"));

if (User.GetClaimValue("http://schemas.microsoft.com/ws/2008/06/identity/claims/role").ToLower().Contains("everyone"))

{

raisUser.Role = UserRole.Normal;

}

else

{

raisUser.Role = UserRole.Admin;

}

}

LocalUtils.WriteLog(User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName"));

if (!User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().Contains("conedinternal") &&

!User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").ToLower().Contains("onrinternal"))

{

strGroups = User.GetClaimValue("http://coned.xmlsoap.org/identity/claims/groupName").Split(',');

raisUser.MarketerIds = new string[strGroups.Length];

for (int i = 0; i < strGroups.Length; i++)

{

strGroupDetails = strGroups[i].Split('\_');

raisUser.MarketerIds[i] = strGroupDetails[2];

}

}

LocalUtils.WriteLog(User.GetClaimValue("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier"));

raisUser.UserName = User.GetClaimValue("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier");

//schemas.xmlsoap.org/ws/2005/05/identity/claims/privatepersonalidentifier - Last Name

}

catch (Exception ex)

{

return View();

//LocalUtils.LogError(ex);

//throw ex;

}

raisUser.IsValid = true;

Session["User"] = raisUser;

}

else

{

return View("../Account/LoginFail");

}

if (raisUser != null && IsBoth == true)

return View("../Main/Choice");

else if (raisUser != null)

{

ViewData["NoTitle"] = true;

return View(raisUser);

}

else

return View("../Account/Login");

}

else

{

raisUser = (RAISUser)Session["User"];

return View();

}

}

[AllowAnonymous]

public ActionResult LoginFail()

{

var authenticationTypes = new string[] {

DefaultAuthenticationTypes.ApplicationCookie,

DefaultAuthenticationTypes.ExternalCookie

};

// AuthenticationManager.SignOut(authenticationTypes);

string[] myCookies = Request.Cookies.AllKeys;

foreach (string cookie in myCookies)

{

Response.Cookies[cookie].Expires = DateTime.Now.AddDays(-1);

}

Response.Cache.SetExpires(DateTime.UtcNow.AddMinutes(-1));

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.Cache.SetNoStore();

Session.RemoveAll();

Session.Clear();

return View();

}

public ActionResult LogOut()

{

var authenticationTypes = new string[] {

DefaultAuthenticationTypes.ApplicationCookie,

DefaultAuthenticationTypes.ExternalCookie

};

//AuthenticationManager.SignOut(authenticationTypes);

string[] myCookies = Request.Cookies.AllKeys;

foreach (string cookie in myCookies)

{

Response.Cookies[cookie].Expires = DateTime.Now.AddDays(-1);

}

Response.Cache.SetExpires(DateTime.UtcNow.AddMinutes(-1));

Response.Cache.SetCacheability(HttpCacheability.NoCache);

Response.Cache.SetNoStore();

Session.RemoveAll();

Session.Clear();

ViewBag.LogOutMessage = "Logged out successfully";

return View();

}

}

}

**Base Controller – handles common actions**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace RAISWeb.Controllers

{

public class BaseController : Controller

{

protected override void OnActionExecuting(ActionExecutingContext filterContext)

{

ViewBag.LogOutURL = ConfigurationManager.AppSettings["LogOutURL"];

}

}

}

**Admin Controller – handles admin actions**

public class AdminController : BaseController

{

public ActionResult Manage()

{

return View();

}

public ActionResult ManageAlerts()

{

ViewBag.Homelink1 = "../Admin/Manage";

ViewBag.HomeText1 = "Admin";

HttpResponseMessage response;

string strResponse = string.Empty;

RAISUser raisUser = (RAISUser)Session["User"];

response = LocalUtils.GetWebApiResponse("AlertsApi?Comp=" + raisUser.Company);

List<Alert> lstAlerts = new List<Alert>();

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstAlerts = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Alert>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

//var model = alertModel.AlertList;

return View(lstAlerts);

}

//[HttpPost]

//[ValidateAntiForgeryToken]

public ActionResult Upload(FormCollection form)

{

ViewBag.Homelink1 = "../Admin/Manage";

ViewBag.HomeText1 = "Admin";

if (Request.Form["Submit"] != null && Request.Files.Count > 0)

{

string FileCategory = form["ddFileCategory"];

string txtDescription = form["txtDescription"];

RAISUser raisUser = (RAISUser)Session["User"];

FileTemplates fileTemplate = new FileTemplates();

Stream fileStream = Request.Files[0].InputStream;

byte[] fileByte = null;

using (var memoryStream = new MemoryStream())

{

fileStream.CopyTo(memoryStream);

fileByte = memoryStream.ToArray();

}

string webApiURL = ConfigurationManager.AppSettings["WebApiURL"];

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(webApiURL);

fileTemplate.Category = FileCategory;

fileTemplate.FileName = Request.Files[0].FileName;

fileTemplate.FileByteArray = fileByte;

fileTemplate.FileDesc = txtDescription;

fileTemplate.Company = raisUser.Company;

var response = client.PostAsync("FileTemplatesApi", fileTemplate, new JsonMediaTypeFormatter()).Result;

if (response.IsSuccessStatusCode)

{

ViewBag.Ack = "File uploaded successfully";

}

else

{

ViewBag.Ack = "File was not uploaded, please try again";

//CommonUtils.LogException(response);

}

}

List<FileTemplates> fileTemplatesModel = GetFileTemplatesForDropDown();

ViewData["FileTemplates"] = new SelectList(fileTemplatesModel, "TemplateId", "TemplateName", FileCategory);

return View();

}

else if (Request.Form["Cancel"] != null)

{

return View("../Admin/Manage");

}

else

{

List<FileTemplates> fileTemplatesModel = GetFileTemplatesForDropDown();

ViewData["FileTemplates"] = new SelectList(fileTemplatesModel, "TemplateId", "TemplateName", -1);

if (Request.Form["Submit"] != null && Request.Files.Count <= 0)

{

ViewBag.Ack = "No file was selected to upload";

}

return View("../Admin/Upload");

}

}

private List<FileTemplates> GetFileTemplatesForDropDown()

{

RAISUser raisUser = (RAISUser)Session["User"];

HttpResponseMessage response;

string strResponse = string.Empty;

if (raisUser.Company == ConEdison.RAISWeb.Interfaces.UserCompany.Coned)

{

response = LocalUtils.GetWebApiResponse("FileTemplatesApi/GetFileTemplates");

}

else

{

response = LocalUtils.GetWebApiResponse("FileTemplatesApi/GetFileTemplatesONR");

}

List<FileTemplates> lstFileTemplates = new List<FileTemplates>();

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstFileTemplates = Newtonsoft.Json.JsonConvert.DeserializeObject<List<FileTemplates>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return lstFileTemplates;

}

public ActionResult EditingInline\_Read([DataSourceRequest] DataSourceRequest request)

{

ViewBag.Homelink1 = "../Admin/Manage";

ViewBag.HomeText1 = "Admin";

List<Alert> lstAlert = new List<Alert>();

return Json(lstAlert.ToDataSourceResult(request), JsonRequestBehavior.AllowGet);

}

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult EditingInline\_Create([DataSourceRequest] DataSourceRequest request, Alert alert)

{

ViewBag.Homelink1 = "../Admin/Manage";

ViewBag.HomeText1 = "Admin";

//string strResponse = string.Empty;

string webApiURL = ConfigurationManager.AppSettings["WebApiURL"];

//List<Alert> lstAlerts = new List<Alert>();

RAISUser raisUser = (RAISUser)Session["User"];

if (string.IsNullOrEmpty(alert.Title) || string.IsNullOrEmpty(alert.Description) || string.IsNullOrEmpty(alert.DatePosted))

{

ViewBag.Ack = "Please enter data in all fields";

return RedirectToAction("ManageAlerts", "Admin");

}

alert.Company = raisUser.Company;

if (alert != null && ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(webApiURL);

var response = client.PostAsync("AlertsApi/AddAlert", alert, new JsonMediaTypeFormatter()).Result;

if (response.IsSuccessStatusCode)

{

//strResponse = response.Content.ReadAsStringAsync().Result;

//lstAlerts = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Alert>>(strResponse);

}

else

{

ViewBag.Ack = "Please try again";

//CommonUtils.LogException(response);

}

}

}

return Json(new[]{alert}.ToDataSourceResult(request, ModelState));

//return View("../Admin/ManageAlerts", lstAlerts);

}

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult EditingInline\_Update([DataSourceRequest] DataSourceRequest request, Alert alert)

{

ViewBag.Homelink1 = "../Admin/Manage";

ViewBag.HomeText1 = "Admin";

//string strResponse = string.Empty;

string webApiURL = ConfigurationManager.AppSettings["WebApiURL"];

//List<Alert> lstAlerts = new List<Alert>();

RAISUser raisUser = (RAISUser)Session["User"];

if (string.IsNullOrEmpty(alert.Title) || string.IsNullOrEmpty(alert.Description) || string.IsNullOrEmpty(alert.DatePosted))

{

ViewBag.Ack = "Please enter data in all fields";

return RedirectToAction("ManageAlerts", "Admin");

}

alert.Company = raisUser.Company;

if (alert != null && ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(webApiURL);

var response = client.PostAsync("AlertsApi/EditAlert", alert, new JsonMediaTypeFormatter()).Result;

if (response.IsSuccessStatusCode)

{

//strResponse = response.Content.ReadAsStringAsync().Result;

//lstAlerts = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Alert>>(strResponse);

}

else

{

ViewBag.Ack = "Please try again";

//CommonUtils.LogException(response);

}

}

}

return Json(new[] { alert }.ToDataSourceResult(request, ModelState));

// Response.Redirect(Request.RawUrl, true);

// RedirectToAction("ManageAlerts", "Admin");

// return null;

// return View("../Admin/ManageAlerts", lstAlerts);

//return View("../Admin/ManageAlerts", lstAlerts);

}

[AcceptVerbs(HttpVerbs.Post)]

public ActionResult EditingInline\_Destroy([DataSourceRequest] DataSourceRequest request, Alert alert)

{

ViewBag.Homelink1 = "../Admin/Manage";

ViewBag.HomeText1 = "Admin";

//string strResponse = string.Empty;

string webApiURL = ConfigurationManager.AppSettings["WebApiURL"];

//List<Alert> lstAlerts = new List<Alert>();

RAISUser raisUser = (RAISUser)Session["User"];

alert.Company = ConEdison.RAISWeb.Interfaces.UserCompany.Coned;

if (alert != null && ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(webApiURL);

var response = client.PostAsync("AlertsApi/DeleteAlert", alert, new JsonMediaTypeFormatter()).Result;

if (response.IsSuccessStatusCode)

{

//strResponse = response.Content.ReadAsStringAsync().Result;

//lstAlerts = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Alert>>(strResponse);

}

else

{

ViewBag.Ack = "Please try again";

//CommonUtils.LogException(response);

}

}

}

return Json(new[] { alert }.ToDataSourceResult(request, ModelState));

//return View("../Admin/ManageAlerts", lstAlerts);

}

}

}

**Bill Controllers – handles billing and invoice process**

public class BillController : BaseController

{

private Invoice invoice = null;

public ActionResult ProcessBilling(FormCollection form)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

string ddMarketer = form["ddComp"];

string selectedText = form["ddSelectedMarketer"];

string txtAccountNumber = form["txtAccountNumber"];

if(string.IsNullOrEmpty(ddMarketer))

{

ddMarketer = Request.QueryString["MarketerId"];

}

if (string.IsNullOrEmpty(selectedText))

{

selectedText = Request.QueryString["MText"];

}

if (Request.Form["btnAcctSummary"] != null)

{

Account account = GetAccountSummary(ddMarketer);

RAISUser raisUser = (RAISUser)Session["User"];

List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

ViewData["Marketers"] = new SelectList(marketerModel, "MarketerId", "MarketerName", 0);

ViewBag.MarketerText = selectedText;

return View("../Bill/AcctSummary", account);

}

else if (Request.Form["btnESCOInvoice"] != null || Request.QueryString["Source"]!=null)

{

List<ESCOInvoice> lstESCOInvoices = GetEscoInvoices(ddMarketer);

RAISUser raisUser = (RAISUser)Session["User"];

List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

ViewData["Marketers"] = new SelectList(marketerModel, "MarketerId", "MarketerName", 0);

ViewBag.MarketerText = selectedText;

ViewBag.MarketrNo = ddMarketer;

return View("../Bill/ESCOInvoice", lstESCOInvoices);

}

else if (Request.Form["btnBillHist"] != null)

{

Account account = new Account();

account = GetBillHist(txtAccountNumber);

return View("../Bill/BillHist", account);

}

else

{

return View("../Main/Billing");

}

}

public Account GetAccountSummary(string MarketerId)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

HttpResponseMessage response = LocalUtils.GetWebApiResponse("MarketerApi/GetAccountSummary?marketerNo=" + MarketerId + "");

Account account = null;

if (response.IsSuccessStatusCode)

{

string strResponse = response.Content.ReadAsStringAsync().Result;

account = Newtonsoft.Json.JsonConvert.DeserializeObject<Account>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return account;

}

public List<ESCOInvoice> GetEscoInvoices(string MarketerId)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

HttpResponseMessage response = LocalUtils.GetWebApiResponse("MarketerApi/GetESCOInvoices?marketerNo=" + MarketerId + "");

List<ESCOInvoice> lstESCOInvoices = null;

if (response.IsSuccessStatusCode)

{

string strResponse = response.Content.ReadAsStringAsync().Result;

lstESCOInvoices = Newtonsoft.Json.JsonConvert.DeserializeObject<List<ESCOInvoice>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return lstESCOInvoices;

}

public ActionResult PrintInvoiceDetails(string InvoiceNo)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

//string InvoiceNo = form["hidddenInvoiceNo"];

HttpResponseMessage response = LocalUtils.GetWebApiResponse("MarketerApi/GetESCOInvoiceDetails?InvoiceNo=" + InvoiceNo + "");

if (response.IsSuccessStatusCode)

{

string strResponse = response.Content.ReadAsStringAsync().Result;

invoice = Newtonsoft.Json.JsonConvert.DeserializeObject<Invoice>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

List<ReportParameter> reportparams = new List<ReportParameter>();

ReportViewer ReportViewer1 = new ReportViewer();

reportparams.Add(new ReportParameter("MarketerCSSName", invoice.MarketerCSSName));

reportparams.Add(new ReportParameter("MarketerAddressLine1", invoice.MarketerAddressLine1));

reportparams.Add(new ReportParameter("MarketerAddressLine2", invoice.MarketerAddressLine2));

reportparams.Add(new ReportParameter("MarketerAddressLine3", invoice.MarketerAddressLine3));

reportparams.Add(new ReportParameter("CssAccountNumber", invoice.CssAccountNumber));

reportparams.Add(new ReportParameter("ServiceClassRate", invoice.ServiceClassRate));

reportparams.Add(new ReportParameter("RepresentativeName", invoice.RepresentativeName));

reportparams.Add(new ReportParameter("BillFromDate", invoice.BillFromDate));

reportparams.Add(new ReportParameter("BillToDate", invoice.BillToDate));

reportparams.Add(new ReportParameter("PreviousBalance", invoice.PreviousBalance));

reportparams.Add(new ReportParameter("LatePaymentCharge", invoice.LatePaymentCharge));

reportparams.Add(new ReportParameter("InvoiceBalance", invoice.InvoiceBalance));

reportparams.Add(new ReportParameter("AmountDue", invoice.AmountDue));

reportparams.Add(new ReportParameter("TotalAdjustmentAndFees", invoice.TotalAdjustmentAndFees));

reportparams.Add(new ReportParameter("TotalThisMonth", invoice.TotalThisMonth));

reportparams.Add(new ReportParameter("InvoiceDate", invoice.InvoiceDate));

reportparams.Add(new ReportParameter("InvoiceDueDate", invoice.InvoiceDueDate));

ReportDataSource rds = new ReportDataSource("LineItems", invoice.LineItemsInvoice);

ReportViewer1.ProcessingMode = ProcessingMode.Local;

ReportViewer1.LocalReport.ReportPath = Server.MapPath("../InvoiceViewer/Invoice.rdlc");

ReportViewer1.LocalReport.DataSources.Clear();

ReportViewer1.LocalReport.DataSources.Add(rds);

ReportViewer1.LocalReport.EnableExternalImages = true;

ReportViewer1.LocalReport.SetParameters(reportparams);

ReportViewer1.LocalReport.Refresh();

byte[] streamBytes = null;

string mimeType = "";

string encoding = "";

string filenameExtension = "";

string[] streamids = null;

Warning[] warnings = null;

streamBytes = ReportViewer1.LocalReport.Render("PDF", null, out mimeType, out encoding, out filenameExtension, out streamids, out warnings);

return File(streamBytes, mimeType, InvoiceNo.ToString());

}

public ActionResult GetInvoiceDetails(string InvoiceNo, string MarketrNo, string MText)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

HttpResponseMessage response = LocalUtils.GetWebApiResponse("MarketerApi/GetESCOInvoiceDetails?InvoiceNo=" + InvoiceNo + "");

if (response.IsSuccessStatusCode)

{

string strResponse = response.Content.ReadAsStringAsync().Result;

invoice = Newtonsoft.Json.JsonConvert.DeserializeObject<Invoice>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

ViewBag.InvoiceNo = InvoiceNo;

ViewBag.MarketrNo = MarketrNo;

ViewBag.MarketerText = MText;

ViewBag.PageTitle = "Invoice";

return View("../Bill/InvoiceDetails", invoice);

}

[HttpGet]

public JsonResult GetData([DataSourceRequest]DataSourceRequest request)

{

List<FileAccounts> lstFiles = new List<FileAccounts>();

return Json(lstFiles.ToDataSourceResult(request), JsonRequestBehavior.AllowGet);

}

public Account GetBillHist(string AccountNumber)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

HttpResponseMessage response = LocalUtils.GetWebApiResponse("MarketerApi/GetBillingHistory?accountNumber=" + AccountNumber + "");

Account account = null;

if (response.IsSuccessStatusCode)

{

string strResponse = response.Content.ReadAsStringAsync().Result;

account = Newtonsoft.Json.JsonConvert.DeserializeObject<Account>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return account;

}

public ActionResult ManHrPrice(FormCollection form)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

if (Request.Form["btnGoBack"] != null)

{

RAISUser raisUser = (RAISUser)Session["User"];

List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

return View("../Main/Billing", marketerModel);

}

else

{

string AccountNumber = form["hdAccountNumber"];

HttpResponseMessage response = LocalUtils.GetWebApiResponse("MarketerApi/GetManHrPricingHistory?accountNumber=" + AccountNumber + "");

List<ManHrPrice> lstManHrPrice = null;

if (response.IsSuccessStatusCode)

{

string strResponse = response.Content.ReadAsStringAsync().Result;

lstManHrPrice = Newtonsoft.Json.JsonConvert.DeserializeObject<List<ManHrPrice>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

ViewBag.AccountNumber = AccountNumber;

return View(lstManHrPrice);

}

}

[HttpGet]

public JsonResult GetManHrPrice([DataSourceRequest]DataSourceRequest request)

{

ViewBag.Homelink1 = "../Main/Billing";

ViewBag.HomeText1 = "Billing";

List<ManHrPrice> lstFiles = new List<ManHrPrice>();

return Json(lstFiles.ToDataSourceResult(request), JsonRequestBehavior.AllowGet);

}

}

**Download Controller – handles generation and downloading of reports**

public class DownloadController : BaseController

{

[HttpPost]

public ActionResult ProcessDownload(FormCollection form)

{

ViewBag.Homelink1 = "../Main/Download";

ViewBag.HomeText1 = "Download";

HttpResponseMessage response;

List<FileAccounts> lstFiles = new List<FileAccounts>();

RAISFile raisFile = new RAISFile();

string contentType = string.Empty;

string strResponse = string.Empty;

byte[] filedata = null;

string ddMarketer = form["ddMarketer"];

string selectedText = form["ddSelectedMarketer"];

ViewBag.Marketer = ddMarketer;

ViewBag.MarketerText = selectedText;

if (Request.Form["btnPreEnroll"] != null)

{

response = LocalUtils.GetWebApiResponse("DownloadApi/GetPreEnrolResultFile?UID=1&Mktr\_CSS\_No=" + ddMarketer + "&blnGeneral=true&strType=P");

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstFiles = Newtonsoft.Json.JsonConvert.DeserializeObject<List<FileAccounts>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return View("../Main/PreEnrol", lstFiles);

}

else if (Request.Form["btnDailyAccount"] != null)

{

response = LocalUtils.GetWebApiResponse("DownloadApi/GetDailyAccountListing?UID=1&Mktr\_CSS\_No=" + ddMarketer + "&blnGeneral=true&strType=P");

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstFiles = Newtonsoft.Json.JsonConvert.DeserializeObject<List<FileAccounts>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return View("../Main/DailyAccount", lstFiles);

}

else if (Request.Form["btnDailyProfile"] != null)

{

response = LocalUtils.GetWebApiResponse("DownloadApi/GetDailyProfileReading?UID=1&Mktr\_CSS\_No=" + ddMarketer + "&blnGeneral=true&strType=P");

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstFiles = Newtonsoft.Json.JsonConvert.DeserializeObject<List<FileAccounts>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return View("../Main/DailyProfile", lstFiles);

}

else if (Request.Form["btnAccountListing"] != null)

{

response = LocalUtils.GetWebApiResponse("DownloadApi/GetONRDailyAccountListing?UID=1&Mktr\_CSS\_No=" + ddMarketer + "&blnGeneral=true&strType=P");

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstFiles = Newtonsoft.Json.JsonConvert.DeserializeObject<List<FileAccounts>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return View("../Main/ONRDailyAccount", lstFiles);

}

else if (Request.Form["btnMiscellaneous"] != null)

{

List<FileTemplates> fileList = GetONRFileLayoutFilesInfo();

return View("../Main/Miscellaneous", fileList);

}

else if (Request.Form["btnPike"] != null)

{

response = LocalUtils.GetWebApiResponse("DownloadApi/GetPikeFile");

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

raisFile = Newtonsoft.Json.JsonConvert.DeserializeObject<RAISFile>(strResponse);

filedata = raisFile.FileByte;

contentType = raisFile.MimeType;

var cd = new System.Net.Mime.ContentDisposition

{

FileName = raisFile.Name,

Inline = true,

};

Response.AppendHeader("Content-Disposition", cd.ToString());

}

else

{

//CommonUtils.LogException(response);

}

return File(filedata, contentType);

}

else if (Request.Form["btnWrongMarketers"] != null)

{

return ExportONRWrongMarketersView();

}

else if (Request.Form["btnTODRSReports"] != null)

{

RAISUser raisUser = (RAISUser)Session["User"];

List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

ViewData["Marketers"] = new SelectList(marketerModel, "MarketerNo", "MarketerName", 0);

return View("../TODRSReports/Download");

}

else

{

response = LocalUtils.GetWebApiResponse("DownloadApi/GetCustomerLeadList?UID=1&Mktr\_CSS\_No=" + ddMarketer + "&blnGeneral=true&strType=P");

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

raisFile = Newtonsoft.Json.JsonConvert.DeserializeObject<RAISFile>(strResponse);

filedata = raisFile.FileByte;

contentType = raisFile.MimeType;

var cd = new System.Net.Mime.ContentDisposition

{

FileName = raisFile.Name,

Inline = true,

};

Response.AppendHeader("Content-Disposition", cd.ToString());

}

else

{

//CommonUtils.LogException(response);

}

return File(filedata, contentType);

}

}

[HttpGet]

//[ValidateAntiForgeryToken]

public ActionResult DownloadFile(string FileId)

{

ViewBag.Homelink1 = "../Main/Download";

ViewBag.HomeText1 = "Download";

RAISUser raisUser = (RAISUser)Session["User"];

HttpResponseMessage response = null;

if (raisUser.Company == ConEdison.RAISWeb.Interfaces.UserCompany.Coned)

{

response = LocalUtils.GetWebApiResponse("DownloadApi/DownloadFile?FileId=" + FileId);

}

else

{

response = LocalUtils.GetWebApiResponse("DownloadApi/DownloadFileONR?FileId=" + FileId);

}

string strResponse = string.Empty;

RAISFile raisFile = new RAISFile();

string contentType = string.Empty;

byte[] filedata = null;

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

raisFile = Newtonsoft.Json.JsonConvert.DeserializeObject<RAISFile>(strResponse);

filedata = raisFile.FileByte;

contentType = raisFile.MimeType;

}

else

{

//CommonUtils.LogException(response);

}

return File(filedata, contentType, raisFile.Name);

}

[HttpGet]

public JsonResult GetData([DataSourceRequest]DataSourceRequest request)

{

ViewBag.Homelink1 = "../Main/Download";

ViewBag.HomeText1 = "Download";

List<FileAccounts> lstFiles = new List<FileAccounts>();

return Json(lstFiles.ToDataSourceResult(request), JsonRequestBehavior.AllowGet);

}

private List<FileTemplates> GetONRFileLayoutFilesInfo()

{

ViewBag.Homelink1 = "../Main/Download";

ViewBag.HomeText1 = "Download";

//RAISUser raisUser = (RAISUser)Session["User"];

HttpResponseMessage response;

string strResponse = string.Empty;

response = LocalUtils.GetWebApiResponse("FileTemplatesApi/GetFileTemplatesONR");

List<FileTemplates> lstFiles = new List<FileTemplates>();

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstFiles = Newtonsoft.Json.JsonConvert.DeserializeObject<List<FileTemplates>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

return lstFiles;

}

public ActionResult ExportONRWrongMarketersView()

{

HttpResponseMessage response;

string strResponse = string.Empty;

List<WrongMarketers> lstWrongMarketers = new List<WrongMarketers>();

response = LocalUtils.GetWebApiResponse("DownloadApi/GetONRWrongMarketers");

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstWrongMarketers = Newtonsoft.Json.JsonConvert.DeserializeObject<List<WrongMarketers>>(strResponse);

}

else

{

//CommonUtils.LogException(response);

}

ReportViewer ReportViewer1 = new ReportViewer();

ReportDataSource rds = new ReportDataSource("dsMarketers", lstWrongMarketers);

ReportViewer1.ProcessingMode = ProcessingMode.Local;

ReportViewer1.LocalReport.ReportPath = Server.MapPath("../InvoiceViewer/WrongMarkters.rdlc");

ReportViewer1.LocalReport.DataSources.Clear();

ReportViewer1.LocalReport.DataSources.Add(rds);

ReportViewer1.LocalReport.EnableExternalImages = true;

//ReportViewer1.LocalReport.SetParameters(reportparams);

ReportViewer1.LocalReport.Refresh();

Warning[] warnings;

string[] streamids;

string mimeType;

string encoding;

string extension;

string filename;

byte[] bytes = ReportViewer1.LocalReport.Render(

"Excel", null, out mimeType, out encoding,

out extension,

out streamids, out warnings);

filename = string.Format("{0}.{1}", "WrongMarketers", "xls");

Response.ClearHeaders();

Response.Clear();

Response.AddHeader("Content-Disposition", "attachment;filename=" + filename);

Response.ContentType = mimeType;

Response.BinaryWrite(bytes);

Response.Flush();

Response.End();

return null;

}

}

**Main Controller – handles main common operations**

public class MainController : BaseController

{

public ActionResult Choice()

{

return View();

}

[HttpPost]

public ActionResult ProcessUserCompany(FormCollection form)

{

RAISUser raisUser = (RAISUser)Session["User"];

if (Request.Form["Coned"] != null)

{

raisUser.Company = UserCompany.Coned;

Session["User"] = raisUser;

ViewData["NoTitle"] = true;

return View("../Home/MainMenu");

}

else

{

raisUser.Company = UserCompany.Oru;

Session["User"] = raisUser;

ViewData["NoTitle"] = true;

return View("../Home/MainMenu");

}

}

public ActionResult Alerts()

{

HttpResponseMessage response;

string strResponse = string.Empty;

RAISUser raisUser = (RAISUser)Session["User"];

response = LocalUtils.GetWebApiResponse("AlertsApi?Comp=" + raisUser.Company);

List<Alert> lstAlerts = new List<Alert>();

if (response.IsSuccessStatusCode)

{

strResponse = response.Content.ReadAsStringAsync().Result;

lstAlerts = Newtonsoft.Json.JsonConvert.DeserializeObject<List<Alert>>(strResponse);

}

else

{

//CommonUtils.LogException(response.Content);

}

return View(lstAlerts);

}

public ActionResult Download()

{

RAISUser raisUser = (RAISUser)Session["User"];

List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

return View(marketerModel);

}

public ActionResult Billing()

{

return View();

}

public ActionResult AccountInvoice()

{

RAISUser raisUser = (RAISUser)Session["User"];

List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

return View(marketerModel);

}

public ActionResult Interval(FormCollection form)

{

RAISUser raisUser = (RAISUser)Session["User"];

List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

ViewData["Marketers"] = new SelectList(marketerModel, "MarketerId", "MarketerName", 0);

return View();

}

public ActionResult PostIntervalData(FormCollection form)

{

string resCode = string.Empty;

string responseString = string.Empty;

RAISUser raisUser = (RAISUser)Session["User"];

List<String> paramsList = new List<string>();

paramsList.Add(raisUser.UserName);

if (raisUser.Company == ConEdison.RAISWeb.Interfaces.UserCompany.Coned)

{

paramsList.Add("C");

}

else

{

paramsList.Add("O");

}

paramsList.Add(form["txtAccountNumber"]);

paramsList.Add(form["txtStartDate"]);

paramsList.Add(form["txtEndDate"]);

paramsList.Add(form["radioInterval"]);

paramsList.Add(form["chkAck"]);

string webApiURL = ConfigurationManager.AppSettings["WebApiURL"];

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(webApiURL);

var response = client.PostAsync("MarketerApi/PostIntervalData", paramsList, new JsonMediaTypeFormatter()).Result;

if (response.IsSuccessStatusCode)

{

responseString = response.Content.ReadAsStringAsync().Result;

resCode = Newtonsoft.Json.JsonConvert.DeserializeObject<string>(responseString);

ViewBag.Ack = "Request submitted sucessfully, queue no is " + resCode;

}

else

{

ViewBag.Ack = "Request wasn't submitted, please try again";

//CommonUtils.LogException(response);

}

}

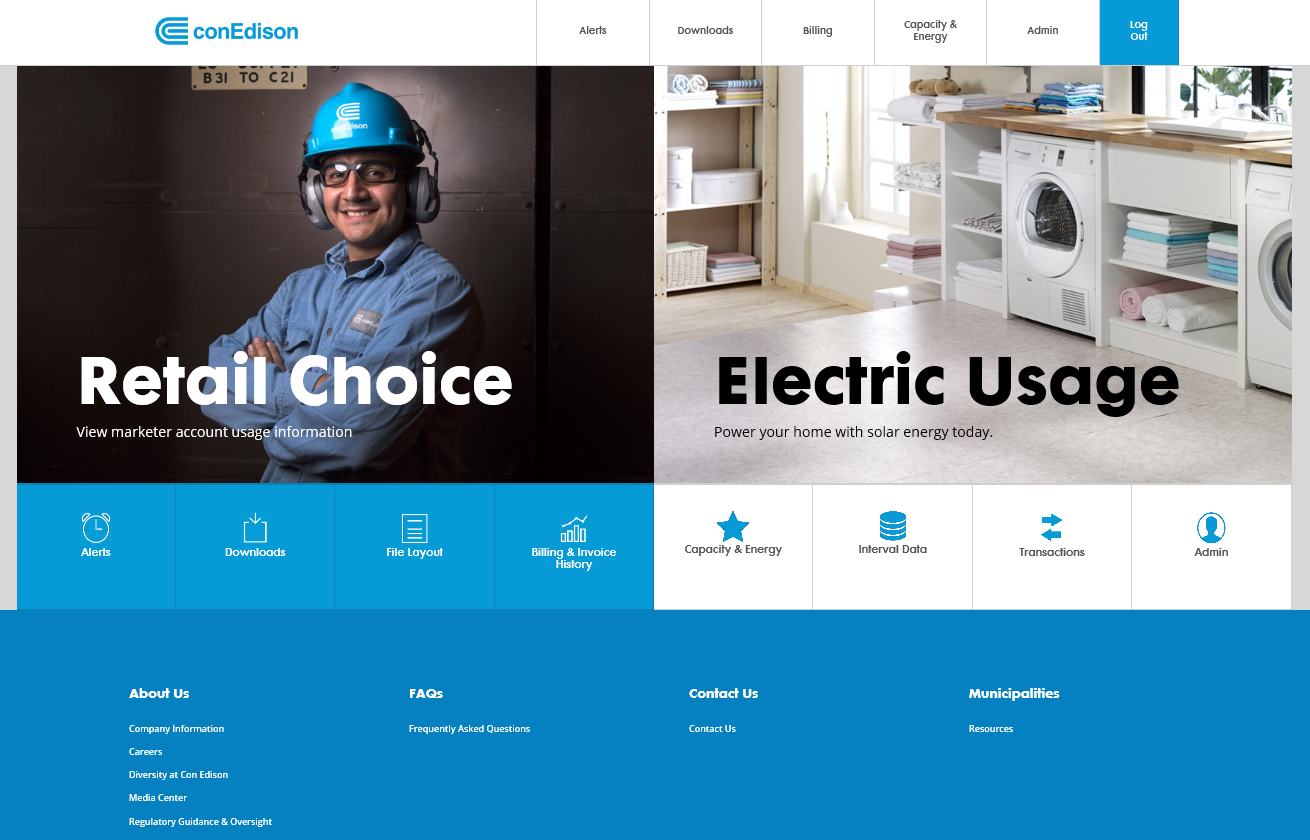
List<Marketer> marketerModel = LocalUtils.GetMarketersForDropDown(raisUser);

ViewData["Marketers"] = new SelectList(marketerModel, "MarketerId", "MarketerName", 0);

return View("../Main/Interval");

}

1. **Application Screen Shots**

****

