```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Web;
using System.Web.Mvc;

using TARS.Helpers;
using TARS.Models;

namespace TARS.Controllers
{
    public class ManagerController : UserController
    {
        protected HistoryDBContext HistoryDB = new HistoryDBContext();

        //
        // Returns manager Index view
        // Overridden from User/Index, which was virtual.
        public override ActionResult Index()
        {
            Authentication auth = new Authentication();
            if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
            {
                return View();
            }
            else
            {
                return View("error");
            }
        }


        //
        /* Retrieves all PCA codes for the specified division and sends them to the view as
         * a list. The division name and a division selection list are also sent to the view.
         */
        public virtual ActionResult searchPCA(string division = null)
        {
            Authentication auth = new Authentication();
            if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
            {
                ViewBag.divisionList = getDivisionSelectList();
                if ( (division == null)||(division.CompareTo("All") == 0) )
                {
                    ViewBag.division = "All";
                    return View(PcaCodeDB.PcaCodeList.ToList());
                }
                else
                {
                    ViewBag.division = division;
                    var pcaList = from p in PcaCodeDB.PcaCodeList
                                  where (p.division.CompareTo(division) == 0)
                                  select p;
                    return View(pcaList.ToList());
                }
            }
            else
            {
                return View("error");
            }
        }


        //
        /* Retrieves all the employee objects of employees that work for the specified departement
         * within the Information Technology division (as per client request), then returns them
         * as a list. If department is null, it displays all employees in the division
         */
        public virtual ActionResult userManagement(DateTime refDate, string department = null)
        {
            Authentication auth = new Authentication();
            if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
            {
                //if it's a redirect from submitRejectTimesheet()
                if (TempData["emailSentFlag"] != null)
                {
                    ViewBag.emailSentFlag = true;
                    ViewBag.messageRecipient = TempData["recipient"];
                }
```

```
        string division = getUserDivision();
        IEnumerable<TARSUser> employees = getDivisionEmployeeObjList(division, department);
        ViewBag.division = division;
        ViewBag.departmentList = getDepartmentSelectList(division);
        ViewBag.refDate = refDate;
        ViewBag.refSunday = refDate.StartOfWeek(DayOfWeek.Sunday);
        ViewBag.refPayPeriod = getPayPeriod(refDate);
        return View(employees);
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves all Work Efforts and sends them to the view as a list, along with a
 * list of PCA Codes associated with each Work Effort. Also, sends variables that
 * the view uses to determine if error messages should be displayed
 */
[HttpGet]
public virtual ActionResult weManagement()
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        var workEffortList = WorkEffortDB.WorkEffortList.ToList();

        //create a list of lists for pca codes
        //(each work effort will have a list of PCA codes)
        ViewBag.pcaListOfLists = new List<List<SelectListItem>>();
        foreach (var item in workEffortList)
        {
            ViewBag.pcaListOfLists.Add(getWePcaCodesSelectList(item));
        }

        //check if an "unable to hide Work Effort error should be displayed"
        if (TempData["failedHide"] != null)
        {
            ViewBag.failedHide = true;
        }
        //check if an "unable to delete Work Effort error should be displayed"
        if (TempData["failedDelete"] != null)
        {
            ViewBag.failedDelete = true;
        }
        return View(workEffortList);
    }
    else
    {
        return View("error");
    }
}


//
//This was attached to delete; not sure what this is yet, but it doesn't explode!
protected override void Dispose(bool disposing)
{
    PcaCodeDB.Dispose();
    base.Dispose(disposing);
}


//
/* Creates an empty Work Effort object and sends it to the view, along with a
 * selection list of divisions.
 */
public virtual ActionResult addWorkEffort()
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        ViewBag.divisionList = getDivisionSelectList();
        return View(new WorkEffort());
    }
    else
    {
        return View("error");
    }
}
```

```
//
/* Receives a Work Effort object and checks to make sure that the dates are valid
 * and that they are within the associated PCA Code's time boundaries.  If so, then
 * it is saved to the database.  If not, then the object is returned to the view
 * for editing.
 */
[HttpPost]
public virtual ActionResult addWorkEffort(WorkEffort workeffort)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        if (ModelState.IsValid)
        {
            //make sure there is an end date
            if (workeffort.endDate == null)
            {
                workeffort.endDate = DateTime.MaxValue;
            }

            //make sure the start date is before the end date
            if (workeffort.startDate > workeffort.endDate)
            {
                ViewBag.divisionList = getDivisionSelectList();
                ViewBag.endBeforeStartFlag = true;
                return View(workeffort);
            }

            //make sure it falls within it's associated PCA code's time boundaries
            if (verifyWeTimeBounds(workeffort, workeffort.pcaCode) == true)
            {
                //update WorkEffort table in database
                WorkEffortDB.WorkEffortList.Add(workeffort);
                WorkEffortDB.SaveChanges();

                //add the PCA_WE association to PCA_WE table
                PCA_WE tmpPcaWe = new PCA_WE();
                tmpPcaWe.WE = workeffort.ID;
                tmpPcaWe.PCA = getPcaIdFromCode(workeffort.pcaCode);
                tmpPcaWe.associationStartDate = DateTime.Now;
                tmpPcaWe.associationEndDate = DateTime.MaxValue;
                tmpPcaWe.active = true;
                PCA_WEDB.PCA_WEList.Add(tmpPcaWe);
                PCA_WEDB.SaveChanges();

                return RedirectToAction("weManagement");
            }
            else
            {
                ViewBag.divisionList = getDivisionSelectList();
                ViewBag.notWithinTimeBounds = true;
                return View(workeffort);
            }
        }
        return View("error");
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves the Work Effort object with specified ID and returns it to the view.
 * If the user is an Admin, ViewBag.adminFlag is set to TRUE.  In the view, if
 * ViewBag.adminFlag is TRUE, then an option to ADD or REMOVE PCA Code associations
 * is displayed.
 */
[HttpGet]
public virtual ActionResult editWorkEffort(int id)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        WorkEffort workeffort = WorkEffortDB.WorkEffortList.Find(id);
        ViewBag.pcaList = getWePcaCodesSelectList(workeffort);

        Authentication newAuth = new Authentication();
        if (newAuth.isAdmin(this))
        {
            ViewBag.adminFlag = true;
```

```
        }
        return View(workeffort);
    }
    else
    {
        return View("error");
    }
}


//
/* Receives a Work Effort object and checks to make sure that the dates are valid
 * and that they are within the associated PCA Code's time boundaries.  If so, then
 * it is saved to the database.  If not, then the object is returned to the view
 * for editing.
 */
[HttpPost]
public virtual ActionResult editWorkEffort(WorkEffort workeffort)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        if (ModelState.IsValid)
        {
            //make sure the start date is before the end date
            if (workeffort.startDate > workeffort.endDate)
            {
                ViewBag.endBeforeStartFlag = true;
                ViewBag.pcaList = getWePcaCodesSelectList(workeffort);
                Authentication newAuth = new Authentication();
                if (newAuth.isAdmin(this))
                {
                    ViewBag.adminFlag = true;
                }
                return View(workeffort);
            }

            //make sure it falls within it's associated PCA code's time boundaries
            if (verifyWeTimeBounds(workeffort, workeffort.pcaCode) == true)
            {
                WorkEffortDB.WorkEffortList.Add(workeffort);
                WorkEffortDB.Entry(workeffort).State = System.Data.EntityState.Modified;
                WorkEffortDB.SaveChanges();
                return RedirectToAction("weManagement");
            }
            else
            {
                ViewBag.notWithinTimeBounds = true;
                ViewBag.pcaList = getWePcaCodesSelectList(workeffort);
                Authentication newAuth = new Authentication();
                if (newAuth.isAdmin(this))
                {
                    ViewBag.adminFlag = true;
                }
                return View(workeffort);
            }
        }
        return View(workeffort);
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves Work Effort object with specified ID and returns it to the view, along
 * with a selection list of PCA Codes associated with it.
 */
[HttpGet]
public virtual ActionResult deleteWorkEffort(int id)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        WorkEffort workeffort = WorkEffortDB.WorkEffortList.Find(id);
        ViewBag.pcaList = getWePcaCodesSelectList(workeffort);
        return View(workeffort);
    }
    else
    {
        return View("error");
```

```
    }
}


//
/* Receives a Work Effort ID and calls a function to check if there are any hours billed to
 * it.  If so, TempData["failedDelete"] is set to TRUE (so an error message will be displayed),
 * and it redirects back to "weManagement". Otherwise, the Work Effort object is retrieved and
 * deleted from the database, and deactivateAllPcaWeForWorkEffort(id) is called to deactivate
 * all of the PCA_We associations for the Work Effort.
 */
[HttpPost, ActionName("deleteWorkEffort")] //This action MUST match the above delete function.
public virtual ActionResult confirmedDeleteWorkEffort(int id)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        //make sure that there aren't any hours billed to the work effort
        if (checkWeForBilledHours(id) == true)
        {
            TempData["failedDelete"] = true;
            return RedirectToAction("weManagement");
        }
        else
        {
            WorkEffort workeffort = WorkEffortDB.WorkEffortList.Find(id);
            //change the active status to FALSE for all PCA_WE entries for the work effort
            deactivateAllPcaWeForWorkEffort(id);
            //delete the work effort
            WorkEffortDB.WorkEffortList.Remove(workeffort);
            WorkEffortDB.SaveChanges();
            return RedirectToAction("weManagement");
        }
    }
    else
    {
        return View("error");
    }
}


//
// Changes active status to FALSE for all PCA_WE entries for the specified Work Effort.
// Also changes "associationEndDate" to the current day and time
public void deactivateAllPcaWeForWorkEffort(int id)
{
    List<PCA_WE> tmpPcaWe = new List<PCA_WE>();
    var searchPcaWe = from p in PCA_WEDB.PCA_WEList
                      where p.WE == id
                      select p;
    tmpPcaWe.AddRange(searchPcaWe);
    foreach (var item in tmpPcaWe)
    {
        item.active = false;
        item.associationEndDate = DateTime.Now;
        //save changes in the database
        PCA_WEDB.Entry(item).State = System.Data.EntityState.Modified;
        PCA_WEDB.SaveChanges();
    }
    return;
}


//
/* Retrieves the Work Effort object with specified ID and checks to see if any
 * hours are currently billed to it in timesheets that are NOT locked yet.
 * Returns TRUE if there are.
 */
public bool checkWeForBilledHours(int id)
{
    Timesheet tmpTimesheet = new Timesheet();

    //get all hours that have been billed to the work effort
    var searchHours = from h in HoursDB.HoursList
                      where h.workEffortID == id
                      select h;
    foreach (var hrs in searchHours)
    {
        //If the corresponding timesheet is still active, return TRUE
        tmpTimesheet = getTimesheet(hrs.creator, hrs.timestamp);
        if (tmpTimesheet.locked == false)
        {
            return true;
```

```
        }
    }
    return false;       //if there are no hours billed to it, return false
}


//
/* Retrieves Work Effort object with specified ID, and changes "hidden" to TRUE
 * if the end date was at least one day ago.  (note: Hidden Work Efforts don't
 * show up for employee's when they are adding a Work Effort to their timesheet,
 * but they do still show up in Manager/weManagement where they can be edited and
 * un-hidden)
 */
public virtual ActionResult hideWorkEffort(int id)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        WorkEffort we = WorkEffortDB.WorkEffortList.Find(id);
        if (we.endDate < DateTime.Today)
        {
            we.hidden = true;
            WorkEffortDB.Entry(we).State = System.Data.EntityState.Modified;
            WorkEffortDB.SaveChanges();
            return RedirectToAction("WeManagement");
        }
        else
        {
            TempData["failedHide"] = true;
            return RedirectToAction("WeManagement");
        }
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves Work Effort object with specified ID, and changes "hidden" to FALSE
 * so users can log hours to it on their timesheet
 */
public virtual ActionResult unHideWorkEffort(int id)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        WorkEffort we = WorkEffortDB.WorkEffortList.Find(id);
        we.hidden = false;
        WorkEffortDB.Entry(we).State = System.Data.EntityState.Modified;
        WorkEffortDB.SaveChanges();

        return RedirectToAction("WeManagement");
    }
    else
    {
        return View("error");
    }
}


//
// Retrieves all PCA_WE objects and sends it to the view as a list
[HttpGet]
public virtual ActionResult searchPCA_WE()
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        return View(PCA_WEDB.PCA_WEList.ToList());
    }
    else
    {
        return View("error");
    }
}


//
// Retrieves PCA_WE object with specified ID and sends it to the view
[HttpGet]
public virtual ActionResult viewPCA_WE(int id)
```

```
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        PCA_WE pca_we = PCA_WEDB.PCA_WEList.Find(id);
        return View(pca_we);
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves a list of the specified user's hours for the time period that tsDate falls within.
 * The list is saved in TempData[], then convertHoursForTimesheetView() is called.
 * convertHoursForTimesheetView() returns a list of TimesheetRow objects, which is sent to the view.
 */
[HttpGet]
public virtual ActionResult approveTimesheet(int userKeyID, DateTime tsDate)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        TARSUser employee = TARSUserDB.TARSUserList.Find(userKeyID);
        Timesheet timesheet = getTimesheet(employee.userName, tsDate);

        if (timesheet == null)
        {
            createTimesheet(employee.userName, DateTime.Now);
            ViewBag.timesheet = getTimesheet(employee.userName, DateTime.Now);
        }
        else
        {
            ViewBag.timesheet = timesheet;
        }

        var hoursList = from m in HoursDB.HoursList
                        where (m.creator.CompareTo(employee.userName) == 0)
                        where m.timestamp >= timesheet.periodStart
                        where m.timestamp <= timesheet.periodEnd
                        select m;

        TempData["hoursList"] = hoursList;
        //convert hoursList into a format that the view can use
        List<TimesheetRow> tsRows = convertHoursForTimesheetView();

        ViewBag.workEffortList = getVisibleWorkEffortSelectList(getUserDivision());
        ViewBag.refDate = tsDate;
        ViewBag.userKeyID = userKeyID;
        return View(tsRows);
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves Timesheet object with specified ID and changes "submitted" and
 * "approved" statuses to TRUE.  Also sends a notification email to the employee
 */
public virtual ActionResult submitApproveTimesheet(int id)
{
    if (id >= 0)
    {
        Authentication auth = new Authentication();
        if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
        {
            Timesheet ts = new Timesheet();
            ts = TimesheetDB.TimesheetList.Find(id);
            ts.submitted = true;
            ts.approved = true;
            TimesheetDB.Entry(ts).State = System.Data.EntityState.Modified;
            //save changes to the database
            TimesheetDB.SaveChanges();

            //send an email to employee to notify of timesheet approval
            string body = "Your IDHW timesheet for the pay period of " + ts.periodStart +
                          " - " + ts.periodEnd + " has been approved by a manager.";
            SendEmail(ts.worker, "Timesheet Approved", body);
```

```
            return RedirectToAction("userManagement", new { refDate = DateTime.Now });
        }
        else
        {
            return View("error");
        }
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves Timesheet object with specified ID and changes "submitted" and
 * "approved" statuses to FALSE.  Also sends a notification email to the employee
 */
public virtual ActionResult submitRejectTimesheet(int id)
{
    if (id >= 0)
    {
        Authentication auth = new Authentication();
        if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
        {
            Timesheet ts = new Timesheet();
            ts = TimesheetDB.TimesheetList.Find(id);
            ts.submitted = false;
            ts.approved = false;
            //save changes to the database
            TimesheetDB.Entry(ts).State = System.Data.EntityState.Modified;
            TimesheetDB.SaveChanges();

            //send an email to employee to notify them
            string body = "Your IDHW timesheet for the pay period of " + ts.periodStart +
                          " - " + ts.periodEnd + " has been rejected by a manager. " +
                          "Please fix it and re-submit as soon as possible.<br /><br />Thanks!";
            SendEmail(ts.worker, "Rejected Timesheet", body);
            TempData["emailSentFlag"] = true;
            TempData["recipient"] = ts.worker;
            TempData["emailError"] = TempData["emailError"];

            return RedirectToAction("userManagement", new { refDate=DateTime.Now });
        }
        else
        {
            return View("error");
        }
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves Hours object with specified ID and sends it to the view, along with the
 * Work Effort that the hours are logged to (so description can be displayed), a
 * time code selection list (so the time code can be edited), and flags that let the
 * view know if the hours are editable.
 */
[HttpGet]
public virtual ActionResult managerEditHours(int hrsID, int tsID)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        Hours hours = HoursDB.HoursList.Find(hrsID);
        WorkEffort we = WorkEffortDB.WorkEffortList.Find(hours.workEffortID);
        Timesheet timesheet = TimesheetDB.TimesheetList.Find(tsID);
        ViewBag.timesheetLockedFlag = timesheet.locked;
        Authentication newAuth = new Authentication();
        bool adminFlag = newAuth.isAdmin(this);
        ViewBag.adminFlag = adminFlag;
        ViewBag.workEffort = we;
        ViewBag.timeCodeList = getTimeCodeList();
        return View(hours);
    }
    else
    {
        return View("error");
```

```
    }
}


//
/* Receives Hours object and saves it to the database as modified, then redirects
 * back to Manager/approveTimesheet
 */
[HttpPost]
public virtual ActionResult managerEditHours(Hours tmpHours)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        if (ModelState.IsValid)
        {
            HoursDB.Entry(tmpHours).State = EntityState.Modified;
            HoursDB.SaveChanges();
        }
        int userKeyID = getUserKeyID(tmpHours.creator);
        return RedirectToAction("approveTimesheet", new { userKeyID = userKeyID, tsDate = tmpHours.timestamp });
    }
    else
    {
        return View("error");
    }
}


//
/* Receives an employee's userName and retrieves the TARSUser object for that
 * userName, then returns the object's unique ID
 */
public int getUserKeyID(string worker)
{
    int userID = 0;
    var searchID = from m in TARSUserDB.TARSUserList
                   where (m.userName.CompareTo(worker) == 0)
                   select m;
    userID = searchID.First().ID;
    return userID;
}


//
/* Receives a 5-digit code and retrieves the PcaCode object with that code,
 * then returns the object's unique ID
 */
public int getPcaIdFromCode(int pcacode)
{
    int pcaID = 0;
    var searchID = from m in PcaCodeDB.PcaCodeList
                   where m.code == pcacode
                   select m;
    foreach (var item in searchID)
    {
        pcaID = item.ID;
    }
    return pcaID;
}


//
// Retrieves the PcaCode object with the specified code and returns the object
public PcaCode getPcaObjFromCode(int pcacode)
{
    PcaCode pcaCodeObj = new PcaCode();
    var searchPca = from m in PcaCodeDB.PcaCodeList
                    where m.code == pcacode
                    select m;
    foreach (var item in searchPca)
    {
        pcaCodeObj = item;
    }
    if (pcaCodeObj != null)
    {
        return pcaCodeObj;
    }
    else
    {
        return null;
    }
}
```

```
//
// Retrieves the pcaCode object with the specified ID and returns the 5-digit code
// (note: called from searchPCA_WE View)
public int getPcaCodeFromID(int id)
{
    PcaCode pcaCodeObj = PcaCodeDB.PcaCodeList.Find(id);
    if (pcaCodeObj != null)
    {
        return pcaCodeObj.code;
    }
    else
    {
        return 0;
    }
}


//
// Retrieves the PcaCode object with specified ID and returns the division name
// (note: called from searchPCA_WE View)
public string getPcaDivisionFromID(int id)
{
    PcaCode pcaCodeObj = PcaCodeDB.PcaCodeList.Find(id);
    if (pcaCodeObj != null)
    {
        return pcaCodeObj.division;
    }
    else
    {
        return null;
    }
}


//
// Retrieves the PcaCode object with specified ID and returns the description
// (note: called from searchPCA_WE View)
public string getPcaDescriptionFromID(int id)
{
    PcaCode pcaCodeObj = PcaCodeDB.PcaCodeList.Find(id);
    if (pcaCodeObj != null)
    {
        return pcaCodeObj.description;
    }
    else
    {
        return null;
    }
}


//
/* Retrieves all PCA Codes for the specified division and returns the result
 * as a selection list
 */
public virtual List<SelectListItem> getDivisionPcaCodeList(string division)
{
    List<SelectListItem> pcaCodesList = new List<SelectListItem>();
    var searchPcaCodes = from m in PcaCodeDB.PcaCodeList
                         select m;
    if (division.CompareTo("All") != 0)
    {
        searchPcaCodes = from m in searchPcaCodes
                         where (m.division.CompareTo(division) == 0)
                         select m;
    }
    foreach (var item in searchPcaCodes)
    {
        pcaCodesList.Add(new SelectListItem
        {
            Text = item.code.ToString() + " (" + item.division + ")",
            Value = item.code.ToString()
        });
    }
    return pcaCodesList;
}


//
/* Retrieves list of employees that work for specified division and department, and returns
 * them as a list.  If division is null, it returns a list of all employees
```

```
 */
public virtual List<TARSUser> getDivisionEmployeeObjList(string division = null, string department = null)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        List<TARSUser> divEmployees = new List<TARSUser>();
        var searchUsers = from m in TARSUserDB.TARSUserList
                          select m;
        if ( (division != null)&&(division.CompareTo("All") != 0) )
        {
            if ( (department != null)&&(department.CompareTo("All") != 0) )
            {
                searchUsers = from m in searchUsers
                              where (m.company.CompareTo(division) == 0)
                              where (m.department.CompareTo(department) == 0)
                              select m;
            }
            else
            {
                searchUsers = from m in searchUsers
                              where (m.company.CompareTo(division) == 0)
                              select m;
            }
        }
        foreach (var item in searchUsers)
        {
            divEmployees.Add(item);
        }
        return divEmployees;
    }
    else
    {
        return null;
    }
}


//
/* Retrieves list of employee that work for specified division and department, and returns
 * the names as a selection list.  If division is null, it returns all employee names
 */
public virtual List<SelectListItem> getDivisionEmployeeSelectList(string division = null)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        List<SelectListItem> employeeNames = new List<SelectListItem>();
        var searchUsers = from m in TARSUserDB.TARSUserList
                          select m;
        if ((division != null) && (division.CompareTo("All") != 0))
        {
            searchUsers = from m in searchUsers
                          where (m.company.CompareTo(division) == 0)
                          select m;
        }

        employeeNames.Add(new SelectListItem { Text = "All", Value = "All" });

        foreach (var item in searchUsers)
        {
            employeeNames.Add(new SelectListItem
            {
                Text = item.userName,
                Value = item.userName
            });
        }
        return employeeNames;
    }
    else
    {
        return null;
    }
}


//
/* Retrieves all departments that have at least one employee in the specified division
 * and returns them as a selection list
 */
public List<SelectListItem> getDepartmentSelectList(string division)
{
    List<SelectListItem> deptSelectList = new List<SelectListItem>();
```

```
    List<string> deptList = new List<string>();
    var searchUsers = from u in TARSUserDB.TARSUserList
                      where (u.company.CompareTo(division) == 0)
                      select u;
    //store list of departments in the division (with duplicates)
    foreach (var item in searchUsers)
    {
        deptList.Add(item.department);
    }
    //remove duplicates
    deptList = deptList.Distinct().ToList();

    deptSelectList.Add(new SelectListItem { Text = "All", Value = "All" });
    foreach (var item in deptList)
    {
        deptSelectList.Add(new SelectListItem
        {
            Text = item,
            Value = item
        });
    }
    return deptSelectList;
}


//
/* Calls getDivisionEmployeeSelectList(division) to get the selection list of employees
 * for the division, then converts it to a Json selection list and returns it
 */
public ActionResult jsonDivisionEmployeeSelectList(string division)
{
    List<SelectListItem> employeeSelectList = getDivisionEmployeeSelectList(division);
    List<string> nameList = new List<string>();
    foreach (var item in employeeSelectList)
    {
        nameList.Add(item.Value);
    }
    return Json(nameList.Select(x => new { value = x, text = x }),
                JsonRequestBehavior.AllowGet
                );
}


//
/* Checks to see if the work effort falls within the specified PCA code's time boundaries.
 * If so, it returns TRUE.
 */
public bool verifyWeTimeBounds(WorkEffort effort, int pca)
{
    var searchPCA = from m in PcaCodeDB.PcaCodeList
                    where (m.code.CompareTo(pca) == 0)
                    select m;
    foreach (var item in searchPCA)
    {
        if ((item.startDate <= effort.startDate) && (item.endDate >= effort.endDate))
        {
            return true;
        }
    }
    return false;
}


//
// Retrieves PcaCode object with specified code and returns its time boundaries as a string
//(note: it's called from addWorkEffort View)
public string getPcaTimeBoundsString(int pcacode)
{
    string bounds = "";
    var searchPCA = from m in PcaCodeDB.PcaCodeList
                    where m.code == pcacode
                    select m;
    foreach (var item in searchPCA)
    {
        bounds = item.startDate + " - " + item.endDate;
    }
    return bounds;
}


//
// Returns all database tables as a selection list
public virtual List<SelectListItem> getDbTableSelectList()
```

```
{
    Authentication auth = new Authentication();
    if (auth.isUser(this) || Authentication.DEBUG_bypassAuth)
    {
        List<SelectListItem> tableSelectList = new List<SelectListItem>();
        var tableList = new List<string>();
        tableList.Add("Timesheets");
        tableList.Add("Hours");
        tableList.Add("WorkEfforts");
        tableList.Add("PcaCodes");
        tableList.Add("PCA_WE");
        tableList.Add("TARSUsers");
        tableList.Add("Holidays");

        tableSelectList.Add(new SelectListItem { Text = "All", Value = "All" });

        foreach (var item in tableList)
        {
            tableSelectList.Add(new SelectListItem
            {
                Text = item,
                Value = item
            });
        }
        return tableSelectList;
    }
    else
    {
        return null;
    }
}


//
/* Retrieves all History objects within the specified time frame and sends them
 * to the view as a list
 */
[HttpGet]
public ActionResult viewHistory(DateTime start, DateTime end)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        var searchHist = from h in HistoryDB.HistoryList
                         where h.timestamp >= start
                         where h.timestamp <= end
                         select h;

        ViewBag.start = start.ToShortDateString();
        ViewBag.end = end.ToShortDateString();
        ViewBag.divisionList = getDivisionSelectList();
        ViewBag.employeeList = getDivisionEmployeeSelectList();
        ViewBag.dbtableList = getDbTableSelectList();
        return View(searchHist.ToList());
    }
    else
    {
        return View("error");
    }
}


//
/* Recieves parameters for refining the history results, and narrows down the list to satisfy
 * those parameters.  Then it sends the results back to the view.
 */
[HttpPost]
public ActionResult viewHistory(DateTime start, DateTime end, string division = null, string un = null, string dbtable = null)
{
    Authentication auth = new Authentication();
    if (auth.isManager(this) || Authentication.DEBUG_bypassAuth)
    {
        IEnumerable<History> searchHist = from h in HistoryDB.HistoryList
                                          where h.timestamp >= start
                                          where h.timestamp <= end
                                          select h;

        if ( (division != null)&&(division.CompareTo("All") != 0) )
        {
            IEnumerable<SelectListItem> names = getDivisionEmployeeSelectList(division);
            List<History> tmpList = new List<History>();
            foreach (var item in names)
            {
```

```
                var name = from h in searchHist
                           where (h.username.CompareTo(item.Value) == 0)
                           select h;
                foreach (var nameObj in name)
                {
                    tmpList.Add(nameObj);
                }
            }
            searchHist = (IEnumerable<History>)tmpList;
        }

        if ((un != null) && (un.CompareTo("All") != 0))
        {
            searchHist = from h in searchHist
                         where (h.username.CompareTo(un) == 0)
                         select h;
        }

        if ( (dbtable != null)&&(dbtable.CompareTo("All") != 0) )
        {
            searchHist = from h in searchHist
                         where (h.dbtable.CompareTo(dbtable) == 0)
                         select h;
        }

        ViewBag.start = start.ToShortDateString();
        ViewBag.end = end.ToShortDateString();
        ViewBag.divisionList = getDivisionSelectList();
        ViewBag.employeeList = getDivisionEmployeeSelectList(division);
        ViewBag.dbtableList = getDbTableSelectList();
        return View(searchHist.ToList());
    }
    else
    {
        return View("error");
    }
}


//
/* Calls getDivisionPcaCodeList(division) to get the selection list of PCA Codes
 * for the division, then converts it to a Json selection list and returns it
 */
public ActionResult jsonPcaSelectList(string division)
{
    IEnumerable<SelectListItem> pcaList = getDivisionPcaCodeList(division);

    return Json(pcaList.Select(x => new { value = x.Value, text = x.Text }),
                JsonRequestBehavior.AllowGet
                );
}


//
// Sends notification emails to managers and admin if a PCA or Work Effort Expires this week
public void notificationOfPcaOrWeExpiration()
{
    DateTime refStart = DateTime.Now.StartOfWeek(DayOfWeek.Sunday);
    DateTime refEnd = refStart.AddDays(7);
    DateTime tmpEndDate = DateTime.Now;
    string body = "";
    string pcas = "";
    var searchPca = from p in PcaCodeDB.PcaCodeList
                    where p.endDate >= refStart
                    where p.endDate < refEnd
                    select p;
    var searchWe = from w in WorkEffortDB.WorkEffortList
                   where w.endDate >= refStart
                   where w.endDate < refEnd
                   select w;
    var searchUsers = from t in TARSUserDB.TARSUserList
                      where t.permission > 1
                      select t;
    foreach (var person in searchUsers)
    {
        foreach (var item in searchPca)
        {
            tmpEndDate = (DateTime)item.endDate;
            body = "Notification of PCA expiration this week. <br /><br />" +
                   "PCA code: " + item.code + " (" + item.division + ") <br />" +
                   "Description: \"" + item.description + "\" <br />" +
                   "End Date: " + tmpEndDate.ToShortDateString();
            SendEmail(person.userName, "PCA expiration notification", body);
```

```
                }
                foreach (var item in searchWe)
                {
                    tmpEndDate = (DateTime)item.endDate;
                    pcas = getWePcaCodesString(item.ID);
                    body = "Notification of Work Effort expiration this week. <br /><br />" +
                            "Work Effort description: \"" + item.description + "\" <br />" +
                            "Associated PCA code(s): " + pcas + "<br />End Date: " +
                            tmpEndDate.ToShortDateString();
                    SendEmail(person.userName, "Work Effort expiration notification", body);
                }
            }
            return;
        }
    }
}
```