```csharp
using System;
using System.Collections.Generic;
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 AdminController : ManagerController
    {
        //
        // Displays the default admin view
        [HttpGet]
        public override ActionResult Index()
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                return View();
            }
            else
            {
                return View("error");
            }
        }


        //
        /* Retrieves all PcaCode objects for the specified division and sends them to the
         * view as a list, along with a division selection list (for narrowing results)
         * and current division name.
         */
        public ActionResult maintainPCA(string division = null)
        {
            Authentication auth = new Authentication();
            if(auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                ViewBag.divisionList = getDivisionSelectList();

                if (TempData["failedPcaDelete"] != null)
                {
                    ViewBag.failedPcaDelete = true;
                }
                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");
            }
        }


        //
        /* Creates an empty PcaCode object and sends it to the view, along with a division
         * selection list (for specifying what division the new PCA Code will be in)
         */
        [HttpGet]
        public virtual ActionResult addPCA()
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                ViewBag.divisionList = getDivisionSelectList();
                return View(new PcaCode());
            }
            else
            {
                return View("error");
            }
        }


        //
        /* Recieves a PcaCode object and saves it to the database.  If the start date is
         * before the end date, or if the the PCA Code already exists in the division
         * and has dates that overlap the dates of the new PCA Code, then an error flag
         * is set and the object is sent back to the view for editing.
         */
        [HttpPost]
        public virtual ActionResult addPCA(PcaCode pcacode)
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                if (ModelState.IsValid)
                {
                    if (pcacode.endDate == null)
                    {
                        pcacode.endDate = DateTime.MaxValue;
                    }
                    //make sure the start date is before the end date
                    if (pcacode.startDate > pcacode.endDate)
                    {
                        ViewBag.endBeforeStartFlag = true;
                        ViewBag.divisionList = getDivisionSelectList();
                        return View(pcacode);
                    }

                    /* Make sure that the dates don't overlap if there is another PCA with the same
                     * code in the same division
                     */
                    if (pcaCheckIfDuplicate(pcacode) == false)
                    {
                        PcaCodeDB.PcaCodeList.Add(pcacode);
                        PcaCodeDB.SaveChanges();
                        return RedirectToAction("maintainPCA");
                    }
                    else
                    {
                        ViewBag.duplicatePcaFlag = true;
                        ViewBag.divisionList = getDivisionSelectList();
                        return View(pcacode);
                    }
                }
                return View(pcacode);
            }
            else
            {
                return View("error");
            }
        }


        //
        /* Retrieves PcaCode object with specified ID and sends it to the view, along with
         * a division selection list (for changing the division that the PCA Code is in).
         */
        [HttpGet]
        public virtual ActionResult editPCA(int id)
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                PcaCode pcacode = PcaCodeDB.PcaCodeList.Find(id);
                ViewBag.divisionList = getDivisionSelectList();
                return View(pcacode);
            }
            else
            {
                return View("error");
            }
        }


        //
        /* Recieves a PcaCode object and saves it to the database.  If the start date is
         * before the end date, then an error flag is set and the object is sent back to
         * the view for editing.
         */
        [HttpPost]
        public virtual ActionResult editPCA(PcaCode pcacode)
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                //make sure startDate is prior to endDate
                if (pcacode.startDate > pcacode.endDate)
                {
                    ViewBag.endBeforeStartFlag = true;
                    ViewBag.divisionList = getDivisionSelectList();
                    return View(pcacode);
                }

                /* Make sure that the dates don't overlap if there is another PCA with the same
                 * code in the same division
                 */
                if (pcaCheckIfDuplicate(pcacode) == true)
                {
                    ViewBag.duplicatePcaFlag = true;
                    ViewBag.divisionList = getDivisionSelectList();
                    return View(pcacode);
                }

                TempData["tmpPcaCode"] = pcacode;
                return RedirectToAction("confirmEditPCA");
            }
            else
            {
                return View("error");
            }
        }


        //
        /* Sends TempData["tmpPcaCode"] (which was assigned in editPCA() before calling this
         * method) to the view, along with a division selection list.
         */
        [HttpGet]
        public virtual ActionResult confirmEditPCA()
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                ViewBag.divisionList = getDivisionSelectList();
                return View(TempData["tmpPcaCode"]);
            }
            else
            {
                return View("error");
            }
        }


        //
        // Receives a PcaCode object and saves it to the database as modified
        [HttpPost]
        public virtual ActionResult confirmEditPCA(PcaCode pcacode)
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {

                if (ModelState.IsValid)
                {
                    PcaCodeDB.PcaCodeList.Add(pcacode);
                    PcaCodeDB.Entry(pcacode).State = System.Data.EntityState.Modified;
                    PcaCodeDB.SaveChanges();
                }
                return RedirectToAction("maintainPCA");
            }
            else
            {
                return View("error");
            }
        }


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

```
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            PcaCode pcacode = PcaCodeDB.PcaCodeList.Find(id);
            return View(pcacode);
        }
        else
        {
            return View("error");
        }
    }


    //
    /* Recieves a PcaCode object ID and retrieves the object, then checks to make sure
     * that no Work Efforts are attached to it before deleting it from the database.  If
     * there are Work Efforts attached, then a flag is set so an error message will be
     * displayed, and the Work Effort is not deleted.
     */
    [HttpPost, ActionName("deletePCA")] //This action MUST match the above delete function.
    public virtual ActionResult confirmDeletePCA(int id)
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            //make sure there are no work efforts attached to the PCA
            if (checkPcaForAttachedWorkEfforts(id) == true)
            {
                TempData["failedPcaDelete"] = true;
                return RedirectToAction("maintainPCA");
            }
            else
            {
                PcaCode pcacode = PcaCodeDB.PcaCodeList.Find(id);
                PcaCodeDB.PcaCodeList.Remove(pcacode);
                PcaCodeDB.SaveChanges();
                return RedirectToAction("maintainPCA");
            }
        }
        else
        {
            return View("error");
        }
    }


    //
    /* Receives a PcaCode object ID, then checks to see if any Work Efforts are currently
     * attached to it. It does so by retrieving all PCA_WE entries with a PCA field that
     * matches the ID, and an "active" field set to TRUE. Returns TRUE if any entries are
     * found.
     */
    public bool checkPcaForAttachedWorkEfforts(int id)
    {
        var searchPcaWe = from p in PCA_WEDB.PCA_WEList
                          where p.PCA == id
                          where p.active == true
                          select p;
        foreach (var item in searchPcaWe)
        {
            //return true if there is an entry in PCA_WE table with a matching PCA id
            if (item != null)
            {
                return true;
            }
            return false;
        }
        return false;
    }


    //
    // Retrieves all TARSUser objects and sends them to the view as a list
    public ActionResult userMaintanence()
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            return View(TARSUserDB.TARSUserList.ToList());
        }
        else
        {
            return View("error");
        }
    }


    //
    // NOT YET IMPLEMENTED
    public ActionResult addUser()
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            return View();
        }
        else
        {
            return View("error");
        }
    }


    //
    // NOT YET IMPLEMETED
    public ActionResult editTARSUSer()
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            return View();
        }
        else
        {
            return View("error");
        }
    }


    //
    // NOT YET IMPLEMENTED
    public ActionResult endDateTARSUSer()
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            return View();
        }
        else
        {
            return View("error");
        }
    }


    //
    /* Receives a PcaCode object and checks if its 5-digit code already exists in the
     * division.  If so, it checks if the start and end dates overlap and returns TRUE
     * if they do, FALSE if they don't.
     */
    public bool pcaCheckIfDuplicate(PcaCode pca)
    {
        bool existsFlag = false;
        var searchPca = from p in PcaCodeDB.PcaCodeList
                        where p.ID != pca.ID
                        where p.code == pca.code
                        where (p.division.CompareTo(pca.division) == 0)
                        select p;
        foreach (var item in searchPca)
        {
            //if the date ranges overlap, then set flag to TRUE
            if ( (pca.startDate <= item.endDate)&&(item.startDate <= pca.endDate) )
            {
                existsFlag = true;
            }
        }
        return existsFlag;
    }


    //
    /* Retrieves WorkEffort object with specified ID and sends the description and
     * ID to the view, along with a division selection list (For selecting the division
     * that the new PCA Code will be from. There is a PCA Code dropdown list in the
     * view that is populated via jQuery when a division is selected)
     */
```

```
[HttpGet]
public virtual ActionResult addPCA_WE(int weID)
{
    Authentication auth = new Authentication();
    if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
    {
        WorkEffort we = WorkEffortDB.WorkEffortList.Find(weID);
        ViewBag.divisionList = getDivisionSelectList();
        ViewBag.workEffortDescription = we.description;
        ViewBag.workEffortId = weID;
        return View();
    }
    else
    {
        return View("error");
    }
}


//
/* Receives a PCA_WE object and saves it to the database if the Work Effort is within the
 * time bounds of the PCA Code.  If the Work Effort is NOT within the time PCA Code's time
 * bounds, a flag is set to display an error message, and the object is sent back to the
 * view for editing.
 */
[HttpPost]
public virtual ActionResult addPCA_WE(PCA_WE pca_we)
{
    Authentication auth = new Authentication();
    if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
    {
        //The view actually passed the PCA code instead of the ID, so set the ID to the correct value
        pca_we.PCA = getPcaIdFromCode(pca_we.PCA);

        WorkEffort effort = WorkEffortDB.WorkEffortList.Find(pca_we.WE);
        ViewBag.workEffortDescription = effort.description;
        PcaCode pcaObj = PcaCodeDB.PcaCodeList.Find(pca_we.PCA);

        //make sure it falls within it's associated PCA code's time boundaries
        if (verifyWeTimeBounds(effort, pcaObj.code) == true)
        {
            //Make sure it's not a duplicate entry before adding to database
            if (checkIfDuplicatePcaWe(pca_we) == false)
            {
                //update PCA_WE table
                PCA_WEDB.PCA_WEList.Add(pca_we);
                PCA_WEDB.Entry(pca_we).State = System.Data.EntityState.Added;
                PCA_WEDB.SaveChanges();
            }
            return RedirectToAction("editWorkEffort", "Manager", new { id = pca_we.WE });
        }
        ViewBag.divisionList = getDivisionSelectList();
        ViewBag.workEffortId = effort.ID;
        ViewBag.outOfPcaTimeBounds = true;
        return View(pca_we);
    }
    else
    {
        return View("error");
    }
}


//
/* Retrieves WorkEffort object for the specified ID and sends the description and ID
 * to the view, along with a list of PCA Codes that are associated with the Work Effort
 */
[HttpGet]
public virtual ActionResult deletePCA_WE(int weID)
{
    Authentication auth = new Authentication();
    if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
    {
        WorkEffort we = WorkEffortDB.WorkEffortList.Find(weID);
        ViewBag.workEffortDescription = we.description;
        ViewBag.workEffortId = weID;
        ViewBag.pcaList = getWePcaCodesSelectList(we);
        return View();
    }
    else
    {
        return View("error");
    }
}


//
/* Receives a PCA_WE object and sets the "active" field to FALSE in the database if
 * the PCA Code is NOT the only PCA Code associated with the Work Effort.  If it is
 * the only remaining PCA Code associated with the Work Effort, then a flag is set to
 * display an error and the object is sent back to the view.
 * Deactivates the PCA_WE object and changes the endDate to the current day
 */
[HttpPost]
public virtual ActionResult deletePCA_WE(PCA_WE pca_we)
{
    Authentication auth = new Authentication();
    if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
    {
        //The view actually passed the PCA code instead of the ID, so set the ID to the correct value
        pca_we.PCA = getPcaIdFromCode(pca_we.PCA);
        int count = 0;

        PCA_WE tmpPcaWe = new PCA_WE();
        var searchPcaWe = from p in PCA_WEDB.PCA_WEList
                          where p.WE == pca_we.WE
                          where p.active == true
                          select p;
        foreach (var item in searchPcaWe)
        {
            //This if-statement will only be true once
            if (item.PCA == pca_we.PCA)
            {
                tmpPcaWe = PCA_WEDB.PCA_WEList.Find(item.ID);
            }
            count++;
        }

        if (count > 1)
        {
            //deactivate and set the end date for the PCA_WE entry
            tmpPcaWe.active = false;
            tmpPcaWe.associationEndDate = DateTime.Now;
            // save changes in database
            PCA_WEDB.Entry(tmpPcaWe).State = System.Data.EntityState.Modified;
            PCA_WEDB.SaveChanges();
            return RedirectToAction("weManagement", "Manager");
        }

        ViewBag.lastPcaFlag = true;
        WorkEffort we = WorkEffortDB.WorkEffortList.Find(pca_we.WE);
        ViewBag.workEffortDescription = we.description;
        ViewBag.workEffortId = we.ID;
        ViewBag.pcaList = getWePcaCodesSelectList(we);
        return View();
    }
    else
    {
        return View("error");
    }
}


//
/* Receives a PCA_WE object and checks if the association already exists in the
 * database. Returns TRUE if does. (note: Called by addPCA_WE() before saving the
 * PCA_WE object)
 */
public bool checkIfDuplicatePcaWe(PCA_WE pcawe)
{
    var searchPcaWe = from p in PCA_WEDB.PCA_WEList
                      where p.WE == pcawe.WE
                      where p.PCA == pcawe.PCA
                      where p.active == true
                      select p;
    foreach (var item in searchPcaWe)
    {
        if (item != null)
        {
            return true;
        }
    }
    return false;
}
```

```
        //
        /* Retrieves all the PCA codes in TARS and returns their 5-digit codes
         * as a list of strings.
         */
        public virtual List<string> getAllPcaCodes()
        {
            List<string> pcaList = new List<string>();
            string tmpPca = "";
            var searchPca = from m in PcaCodeDB.PcaCodeList
                            select m;
            foreach (var item in searchPca)
            {
                tmpPca = item.code.ToString();
                pcaList.Add(tmpPca);
            }
            return pcaList;
        }


        //
        /* Retrieves all holidays that are within the same pay period as the specified
         * reference date. If the search results in an empty list, then FALSE is returned.
         * If there is at least one holiday in the list, then TRUE is returned.
         */
        public bool isHolidayWeek(DateTime refDate)
        {
            DateTime refStart = refDate.StartOfWeek(DayOfWeek.Sunday);
            DateTime refEnd = refStart.AddDays(7);
            var searchHolidays = from h in HolidaysDB.HolidaysList
                                 where h.date >= refStart
                                 where h.date < refEnd
                                 select h;
            foreach (var item in searchHolidays)
            {
                return true;
            }
            return false;
        }


        //
        /* Locks all timesheets that ended more than two days ago.  If there was a holiday during
         * the previous pay period, lockDate will have an extra day added to it.
         * (note: called from /ScheduledJobs/TarsScheduledJobs every Tuesday and Wednesday at 12am)
         */
        public void lockTimesheets()
        {
            DateTime refDate = DateTime.Now.Date;
            DateTime lockDate = DateTime.Now.Date;
            Timesheet tmpTimesheet = new Timesheet();
            List<Timesheet> tsList = new List<Timesheet>();

            // If there was a holiday, allow three days after periodEnd before locking
            if (isHolidayWeek(refDate.AddDays(-7)))
            {
                lockDate = refDate.AddDays(-1);
            }
            else
            {
                lockDate = refDate.AddDays(-2);
            }
            var searchTimesheets = from t in TimesheetDB.TimesheetList
                                   where t.locked != true
                                   where t.periodEnd < lockDate
                                   select t;
            foreach (var item in searchTimesheets)
            {
                tmpTimesheet = item;
                tmpTimesheet.locked = true;
                tsList.Add(tmpTimesheet);
            }
            foreach (var item in tsList)
            {
                //save changes in database
                TimesheetDB.Entry(item).State = System.Data.EntityState.Modified;
                TimesheetDB.SaveChanges();
            }
        }


        //
        /* Retrieves all TARSUser objects of employees that work for the specified departement
         * within the Information Technology division and sends them to the view as a list. If
         * department is null, it retrieves all employees in the division.
         */
        public ActionResult viewTimesheetStatuses(DateTime refDate, string department = null)
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                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 the Timesheet object for the specified employee and reference date,
         * changes "locked", "approved", and "submitted" fields to FALSE, and saves the
         * changes in the database.
         */
        public void adminUnlockTimesheet(string username, DateTime refDate)
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                Timesheet tmpTs = new Timesheet();
                var searchTs = from t in TimesheetDB.TimesheetList
                               where (t.worker.CompareTo(username) == 0)
                               where t.periodStart <= refDate
                               where t.periodEnd >= refDate
                               select t;
                foreach (var item in searchTs)
                {
                    tmpTs = item;
                    tmpTs.locked = false;
                    tmpTs.approved = false;
                    tmpTs.submitted = false;
                }
                //save changes in database
                TimesheetDB.Entry(tmpTs).State = System.Data.EntityState.Modified;
                TimesheetDB.SaveChanges();
            }
            return;
        }


        //
        // Retrieves all of the Holidays objects and sends them to the view.
        public ActionResult viewHolidays()
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                List<Holidays> holidays = HolidaysDB.HolidaysList.ToList();
                return View(holidays);
            }
            else
            {
                return View("error");
            }
        }


        //
        // Creates an empty Holidays object and sends it to the view.
        [HttpGet]
        public virtual ActionResult addHoliday()
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                return View(new Holidays());
            }
            else
            {
```

```
            return View("error");
        }
    }


    //
    // Receives a Holidays object and saves it to the database
    [HttpPost]
    public virtual ActionResult addHoliday(Holidays holiday)
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            HolidaysDB.HolidaysList.Add(holiday);
            HolidaysDB.SaveChanges();
            return RedirectToAction("viewHolidays");
        }
        else
        {
            return View("error");
        }
    }


    //
    // Retrieves the Holidays object with specified ID and sends it to the view
    [HttpGet]
    public virtual ActionResult editHoliday(int id)
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            Holidays holiday = HolidaysDB.HolidaysList.Find(id);
            return View(holiday);
        }
        else
        {
            return View("error");
        }
    }


    //
    // Receives a Holidays object and saves it to the database as modified
    [HttpPost]
    public virtual ActionResult editHoliday(Holidays holiday)
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            HolidaysDB.Entry(holiday).State = System.Data.EntityState.Modified;
            HolidaysDB.SaveChanges();
            return RedirectToAction("viewHolidays");
        }
        else
        {
            return View("error");
        }
    }


    //
    // Retrieves the Holidays object with specified ID and sends it to the view
    [HttpGet]
    public virtual ActionResult deleteHoliday(int id)
    {
        Authentication auth = new Authentication();
        if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
        {
            Holidays holiday = HolidaysDB.HolidaysList.Find(id);
            return View(holiday);
        }
        else
        {
            return View("error");
        }
    }


    //
    /* Retrieves the Holidays object with specified ID and deletes if from the database
     * once the user confirms that they want to delete it
     */
```

```
        [HttpPost, ActionName("deleteHoliday")] //This action MUST match the above delete function.
        public virtual ActionResult confirmDeleteHoliday(int id)
        {
            Authentication auth = new Authentication();
            if (auth.isAdmin(this) || Authentication.DEBUG_bypassAuth)
            {
                Holidays holiday = HolidaysDB.HolidaysList.Find(id);
                HolidaysDB.HolidaysList.Remove(holiday);
                HolidaysDB.SaveChanges();
                return RedirectToAction("viewHolidays");
            }
            else
            {
                return View("error");
            }
        }

    }
}
```