```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Data.Entity;
using CupPlaner.Helpers;

namespace CupPlaner.Controllers
{
    public class TeamController : Controller
    {
        // Database container, has functionalities to connect to the database classes.
        CupDBContainer db = new CupDBContainer();
        ScheduleManager sm = new ScheduleManager();

        // GET: Team/Details/5 - Fetches the details of the class, takes the "id"
parameter to determine the corresponding Team object.
        // Returns a Json object, which contains a copy of the corresponding Team
variables.
        public ActionResult Details(int id)
        {
            try
            {
                Team t = db.TeamSet.Find(id);
                List<object> times = new List<object>();
                List<object> matches = new List<object>();
                // Get time intervals for team
                if (t.TimeIntervals != null)
                {
                    foreach (TimeInterval ti in t.TimeIntervals)
                    {
                        times.Add(new { Id = ti.Id, StartTime = ti.StartTime, EndTime =
ti.EndTime });
                    }
                }
                // Get matches for team
                if (t.Matches.Count > 0)
                {
                    foreach (Match m in t.Matches)
                    {
                        Team team1 = m.Teams.ToList()[0];
                        Team team2 = m.Teams.ToList()[1];
                        matches.Add(new { Id = m.Id, Number = m.Number, StartTime =
m.StartTime, FieldName = m.Field.Name, Team1 = new { name = team1.Name, Id = team1.Id },
Team2 = new { name = team2.Name, Id = team2.Id, } });
                    }
                }

                object obj = new { status = "success", Id = t.Id, Name = t.Name,
TimeIntervals = times, Matches = matches};

                return Json(obj, JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Could not find team",
details = ex.Message }, JsonRequestBehavior.AllowGet);
```

```
            }
        }

        // POST: Team/Create - Tries to create a Team object, with the parameters "name"
and "poolId".
        // Tracks the corresponding Pool the team is to be contained in with the
"poolId".
        // Sets the team name and pool to the parameters (name and corresponding Pool
object).
        // Adds the Team object to the database TeamSet, and saves the changes in the
database.
        // Returns a Json object with a state, indicating whether it succeeded creating
the Team object or not.
        [HttpPost]
        public ActionResult Create(string name, int poolId)
        {
            try
            {
                Pool p = db.PoolSet.Find(poolId);
                int max = p.Division.Pools.Max(x => x.Teams.Count);
                if (p.Teams.Count == max)
                {
                    db.FinalsLinkSet.Add(new FinalsLink() { Division = p.Division,
PoolPlacement = max + 1, Finalstage = max + 1 });
                }
                Team t = db.TeamSet.Add(new Team() { Name = name, Pool = p });
                // Add time intervals and default them to the tournament's time intervals
                foreach (TimeInterval ti in p.Division.Tournament.TimeIntervals)
                {
                    TimeInterval timeinterval = new TimeInterval() { Team = t, StartTime
= ti.StartTime, EndTime = ti.EndTime };
                    db.TimeIntervalSet.Add(timeinterval);
                    t.TimeIntervals.Add(timeinterval);
                }
                //Clear the schedule
                sm.DeleteSchedule(p.Division.Tournament.Id, db);

                db.SaveChanges();

                return Json(new { status = "success", message = "New team added", id =
t.Id }, JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "New team not added",
details = ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }

        // POST: Team/Edit/5 - Tries to edit a Team, determined by the "id" parameter and
"poolId".
        // Edits a Teams name and list of TimeIntervals. Saves the changes to the
database, if succeeded.
        // Returns a Json object with a state, indicating whether it succeeded editing
the Team object or not.
        [HttpPost]
        public ActionResult Edit(int id, string name, int poolId, List<DateTime>
startTimes, List<DateTime> endTimes)
```

```
        {
            try
            {
                List<TimeInterval> tis = new List<TimeInterval>();
                List<TimeInterval> tournytis = new List<TimeInterval>();
                Team t = db.TeamSet.Find(id);
                tournytis = t.Pool.Division.Tournament.TimeIntervals.ToList();

                for (int i = 0; i < startTimes.Count; i++)
                {
                    if(startTimes[i] >= tournytis[i].StartTime && endTimes[i] <=
tournytis[i].EndTime)
                    {
                        tis.Add(new TimeInterval() { StartTime = startTimes[i], EndTime =
endTimes[i] });
                    }
                    else
                    {
                        tis.Add(new TimeInterval() { StartTime = tournytis[i].StartTime,
EndTime = tournytis[i].EndTime });
                    }
                }
                // Remove old time intervals
                db.TimeIntervalSet.RemoveRange(t.TimeIntervals);

                // Clear the schedule
                sm.DeleteSchedule(t.Pool.Division.Tournament.Id, db);

                t.Name = name;
                t.TimeIntervals = tis;

                db.Entry(t).State = EntityState.Modified;
                db.SaveChanges();

                return Json(new { status = "success", message = "Team edited" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Team not edited", details
= ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }

        // POST: Team/Delete/5 - Tries to delete a Team object, determined by the "id".
        // Deletes the Team object and saves to the database, if succeeded.
        // Returns a Json object, indicating whether it succeeded deleting the Team
object or not.
        [HttpPost]
        public ActionResult Delete(int id)
        {
            try
            {
                Team t = db.TeamSet.Find(id);
                Pool p = db.PoolSet.Find(t.Pool.Id);

                // Clear the schedule
                sm.DeleteSchedule(t.Pool.Division.Tournament.Id, db);
```

```
                // Remove dependencies
                foreach (TimeInterval ti in t.TimeIntervals.ToList())
                {
                    t.TimeIntervals.Remove(ti);
                }

                db.MatchSet.RemoveRange(t.Matches);
                db.TeamSet.Remove(t);
                db.SaveChanges();

                return Json(new { status = "success", message = "Team Deleted" },
JsonRequestBehavior.AllowGet);
            }
            catch (Exception ex)
            {
                return Json(new { status = "error", message = "Team not deleted", details
= ex.Message }, JsonRequestBehavior.AllowGet);
            }
        }
    }
}
```