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

namespace CupPlaner.Controllers
{
    public class ScheduleManagerController : Controller
    {

        ScheduleManager sm = new ScheduleManager();
        CupDBContainer db = new CupDBContainer();

        //calls DelteSchedule from the ScheduleManager class and returns a json object to 
tell teh frontend weather or not it was successfull
        public ActionResult DeleteSchedule (int tournamentID)
        {
            try
            {
                sm.DeleteSchedule(tournamentID);
                return Json(new { status = "success"}, JsonRequestBehavior.AllowGet);
            }
            catch (Exception)
            {
                return Json(new { status = "error" }, JsonRequestBehavior.AllowGet);
            }
        }

        // finds the minimum number of fields with the same field size that are needed to 
schedule the turnament by calling MinNumOfFields from the ShceduleManager class and
        // returns a json object with the values to the frontend if successfull or an 
error if something went wrong
        public ActionResult MinNumOfFields (int tournamentID, FieldSize fs)
        {
            try
            {
                Tournament t = db.TournamentSet.Find(tournamentID);
                int numOfAllFields = t.Fields.Count(x => x.Size == fs);
                int numOfMinFields = sm.MinNumOfFields(tournamentID, fs);
                return Json(new { status = "success", FieldsCount = numOfMinFields, 
AllFieldsCount = numOfAllFields }, JsonRequestBehavior.AllowGet);
            }
            catch(Exception)
            {
                return Json(new { status = "error" }, JsonRequestBehavior.AllowGet);
            }
        }

        // tries to schedule the turnament through the scheduleAll function from the 
ScheduleManager class. It calls the algorithm with the minimum number of fields required 
at first
        // and increments this number if the algorithm fails. If the number of fields 
reaches the number of fields available for the tournament and the algorithm fails a json 
object
```

```
        // with an error is returned. if the algorithm is successfull, or the tournament
doesnt contain any fields in the right size, a json object with success is returned.
lastly
        // if an exception is throw it returns a json object with an error.
        public ActionResult Schedule(int tournamentID, FieldSize fs)
        {
            try
            {
                int minNumOfFields = 0;
                Tournament t = db.TournamentSet.Find(tournamentID);
                int numOfFields = t.Fields.Count(x => x.Size == fs);
                if (numOfFields > 0 && t.Divisions.Any(x => x.FieldSize == fs))
                {
                    minNumOfFields = sm.MinNumOfFields(tournamentID, fs);
                    for (int i = minNumOfFields; i <= numOfFields; i++)
                    {
                        if (sm.scheduleAll(tournamentID, fs, i))
                        {
                            return Json(new { status = "success" },
JsonRequestBehavior.AllowGet);
                        }
                    }
                }
                else
                {
                    return Json(new { status = "success" },
JsonRequestBehavior.AllowGet);
                }
                return Json(new { status = "error", errorCode = 1, message = "algoritmen
fandt ingen løsning", MinNumOfFields = minNumOfFields }, JsonRequestBehavior.AllowGet);
                }
            catch (Exception)
            {
                return Json(new { status = "error", errorCode = 2, message = "fejl i
programmet" }, JsonRequestBehavior.AllowGet);
            }
        }
    }
}
```