```css
body {
    padding: 0px;
    height: 100%;
    margin: 0px;

    background: rgba(222, 223, 255, 0.50);
}

html, body {
    height: 100%;
}

.loginDiv {
    right: 2%;
    top: 2%;
    position: absolute;
}

.loginButton {
  border: 1px dotted #999;
  border-radius: 0;
}

.loginText {
    position: absolute;
    right: 5%;
    top: 100%;
}

#frontDiv {
    font-family: Geneva, Tahoma, Verdana, sans-serif;
    text-shadow: 2px 1px 1px rgba(109, 116, 243, 0.75);
    border-radius: 5px;
    border: 2px solid rgba(0, 0, 0, .20);
    margin: 0 auto;
    font-size: 40px;
    text-align: center;
    background: linear-gradient(0deg, rgb(253, 252, 255) 60%, rgb(255, 255, 255) 90%);
    background: white;
}

#frontDiv:hover {
    border: 2px solid rgba(0, 0, 0, .40);
    cursor: pointer;
}

#subFrontDiv {
    font-family: Geneva, Tahoma, Verdana, sans-serif;
    text-shadow: 2px 1px 1px rgba(109, 116, 243, 0.75);
    border-radius: 5px;
    border: 2px solid rgba(0, 0, 0, .20);
    font-size: 25px;
    text-align: center;
    background: linear-gradient(0deg, rgb(253, 252, 255) 60%, rgb(255, 255, 255) 90%);
    width: 240px;
}

#subFrontDescription {
    width: 200px;
    border-top: solid 1px rgba(0, 0, 0, 0.25);
```

```
    margin: 0 auto;
    text-decoration: none;
    font-size: 18px;
    text-shadow: none;
    font-family: sans-serif;
    background: rgba(222, 223, 255, 0.30);
    margin-bottom: 10px;
}

.subDivTable {
    width: 500px;
    margin: 0 auto;

}

#rightSideDiv {
    float: right;
    padding-top: 20px;
}

#leftSideDiv {
    float: left;
    padding-top: 20px;
}


.containerFrontDiv {
    background: rgba(222, 223, 255, 0.50);
    border: 1px solid rgba(0, 0, 0, 0.75);
    width: 80%;
    margin: 0 auto;
    margin-top: 20px;
    padding-top: 20px;
    padding-bottom: 20px;

}

.startDiv {
    width: 500px;
    height: 65px;
}

.startDiv:hover {
    border: 2px solid rgba(0, 0, 0, .80);
    cursor: pointer;
    text-decoration: underline;
}
.listingTable {
    border: 1px solid rgba(167, 167, 167, 0.75);
    width: 100%;
    padding-left: 10px;
}

.displayTable {
    border-top: 1px solid rgba(167, 167, 167, 0.75);
    border-bottom: 1px solid rgba(167, 167, 167, 0.75);
    width: 100%;
    padding-left: 10px;
    font-size: 16px;
}
```

```css
.displayTable td {
    padding: 10px;
    width: 50px;

}

.displayDiv {
    padding: 10px;
    line-height: 200%;
}

.displayTable tr:not(:first-child) {
    border-top: 1px solid rgba(167, 166, 255, 0.35);
}

.displayTable tr {
    background: rgba(222, 223, 255, 0.30);
    width: 100%;
}

.listingTable td {
    padding: 10px;
}

.listingTable tr:first-child {
    font-size: 16px;
}
.listingTable tr:hover td {
    background-color: white;
    cursor: pointer;
}

.subDiv {
    width: 100px;
    height: 200px;
}

.mainDiv {
    min-height: 100%;
    position: relative;
    width: 1000px;
    border-left-style: solid;
    border-width: 1px;
    border-right-style: solid;
    border-color: rgba(0, 0, 0, 0.35);
    box-shadow: 3px 5px 15px #888;
    margin: 0 auto;
    background: white;
}

.bannerDiv {
    border-bottom-style: solid;
    border-width: 1px;
    border-color: rgba(0, 0, 0, 0.35);
    height: 124px;

    background-image: url(/imgs/BannerBackground.png);
    background-repeat: no-repeat;
}
```

```
.bottomDiv {
    bottom: 0;
    position: absolute;
    width: 100%;
    border-top: solid 1px rgba(0, 0, 0, 0.35);
    background: white;

}

#subFrontDiv:hover {
    border: 2px solid rgba(0, 0, 0, .40);
    cursor: pointer;
}

/*----- Tabs -----*/
.tabs {
    width:80%;
    height: 250px;
    display:inline-block;
    margin: 0 auto;
    margin-top: 15px;
    position: relative;
    left: 10%;
}

    /*----- Tab Links -----*/
    /* Clearfix */
    .tab-links:after {
        display:block;
        clear:both;
        content:'';
    }

    .tab-links li {
        margin:0px 40px;
        float:left;
        list-style:none;
        margin-left: -35px;
    }

        .tab-links a {
            padding:9px 15px;
            display:inline-block;
            border-radius:1px 1px 0px 0px;
            background:rgba(109, 116, 243, 0.25);
            font-size:16px;
            font-weight:600;
            color:#4c4c4c;
            border: 1px solid black;
            border-bottom: white;
            transition:all linear 0.15s;
        }

        .tab-links a:hover {
            background:rgba(109, 116, 243, 0.50);
            text-decoration:none;
        }

    li.active a, li.active a:hover {
```

```
        background:#fff;
        color:#4c4c4c;
    }

    /*----- Content of Tabs -----*/
    .tab-content {
        padding: 15px;
        padding-top: 0;
        border-radius:1px;
        background:#fff;
        margin: -10px;
    }

        .tab {
            display:none;
            border: solid 1px black;
        }

        .tab.active {
            display:block;
        }

 /*Overrides bootstraps justify method*/
.nav-justified>li {
    display: table-cell;
    float:left;
}
/*FAQ Li CSS*/
.li-header-size{
    font-size: 15px;
    list-style: none;
    list-style-image: url(/imgs/QuestionMark.png);
    background-repeat: no-repeat;

}

.tableFAQ{

    border: 2px solid rgba(0, 0, 0, 0.20);
    border-radius: 25px;
    box-shadow: 5px 5px 5px rgba(0, 0, 0, 0.10);
    width: 800px;
    display: block;
    margin: 0 auto;
    padding-top: 10px;
    padding-bottom: 10px;

}
.tableDataFAQ{
    width: 450px;
    padding-left: 10px;

    vertical-align: top;

}
.Headline{
    margin-top: 5px;
    font-size: 30px;
    text-align: center;
    color: rgba(0, 0, 0, 0.60);
```

```
    font-family: "Times New Roman", Times, serif;

}
.textColor{
    font-size: 20px;
    color: rgba(0, 0, 0, 0.60);
    font-family: "Times New Roman", Times, serif;
}
.detailFormat{
    padding-left: 20px;
}
.autobiographyFormat{
    text-align: left;
    padding-left: 20px;
    padding-right: 20px;
}
.loginCreate{
    border: 2px solid rgba(0, 0, 0, 0.20);
    border-radius: 25px;
    box-shadow: 5px 5px 5px rgba(0, 0, 0, 0.10);
    padding: 15px;
    display: inline-block;


    margin-left: 20px;

    margin: 20px;

    margin-left: 20px;
    margin: 20px;


}




/* LiveChat CSS sektion */
.chatDiv {
    position: absolute;
    left: 20%;
    width: 60%;
    top: 159px;
    padding-left: 25px;
    padding-right: 25px;
    height: 600px;
    border: 1px solid black;
    border-top: none;
    background: white;
    overflow: auto;
    font-size: 12px;
}

.chatDivTable {
    width: 100%;
    min-height: 100%;
    margin: 0px auto;
    text-align: center;
    border-left: 1px solid rgba(0, 0, 0, 0.20);
    border-right: 1px solid rgba(0, 0, 0, 0.20);
```

```
}

.chatDivTable td:nth-of-type(odd) {
    border-right: 1px solid rgba(0, 0, 0, 0.20);
}

.chatDivTable tr:first-child {
    background: black;
    color: white;
    font-size: 20px;
}

.chatDivTable tr:nth-child(2) {
    font-size: 16px;
}

.chatDivTable td {
    border-bottom: 1px solid rgba(0, 0, 0, 0.20);

    margin-top: 20px;
}

.creatorVotesDiv {
    position: absolute;
    top: 0;
    left: 20%;
    text-align: center;
}

.challengerVotesDiv {
    position: absolute;
    top: 0;
    right: 20%;
    text-align: center;
}

.voteDiv {
    background: black;
    color: white;
    font-size: 18px;
    padding: 5px;
    width: 200px;
}


.nickNameDiv {
    border: 1px solid black;
    background: white;
    font-size: 16px;
}

.timerDiv {
    position: absolute;
    top: 0;
    left: 39%;
    text-align: center;
    font-size: 35px;
}

.descriptionDiv {
```

```
    position: absolute;
    top: 59px;
    left: 20%;
    width: 60%;
    background: white;
    border: 1px solid black;
    height: 100px;
}

.messageDiv {
    border: 1px solid black;
    width: 60%;
    position: fixed;
    bottom: 0;
    height: 85px;
    background: white;
    left: 20%;
    border-bottom: none;
}

.timerButton {

}

.chatTextBox {
    width: 500px;
    font-size: 12px;
    height: 90%;

}

.chatButton {
    height: 100%;
    width: 100px;
    font-size: 14px;
    font-weight: bold;
    border-radius: 2px;
    margin-top: 20px;

}

#spectatorInformation {
    position: fixed;
    left: 0;
    bottom: 0;
    width: 200px;
    border: 1px solid black;
    border-top: none;
    border-bottom: none;
    height: 100%;
    background:rgba(232, 233, 248, 0.50);
    box-shadow: 5px 5px 5px #000000;
    box-shadow: 5px 5px 5px rgba(5, 15, 2, 0.30);
    padding: 2px;
    font-size: 12px;
}

#spectatorChat {
    position: fixed;
    right: 0;
```

```
    bottom: 0;
    width: 200px;
    border: 1px solid black;
    border-top: none;
    border-bottom: none;
    height: 100%;
    background:rgba(232, 233, 248, 0.50);
    box-shadow: 5px 5px 5px #000000;
    box-shadow: 5px 5px 5px rgba(5, 15, 2, 0.30);
    padding: 2px;
    font-size: 12px;
    overflow: auto;
}


body {
    padding-top: 50px;
    padding-bottom: 20px;

}

/* Set padding to keep content from hitting the edges */
.body-content {
    padding-left: 15px;
    padding-right: 15px;
}

/* Set width on the form input elements since they're 100% wide by default */
input,
select,
textarea {
    /*max-width: 280px;*/
}

/* styles for validation helpers */
.field-validation-error {
    color: #b94a48;
}

.field-validation-valid {
    display: none;
}

input.input-validation-error {
    border: 1px solid #b94a48;
}

input[type="checkbox"].input-validation-error {
    border: 0 none;
}

.validation-summary-errors {
    color: #b94a48;
}

.validation-summary-valid {
    display: none;
}
```

```
.Test {
            -moz-box-shadow:inset 0px 1px 0px 0px #bee2f9;
            -webkit-box-shadow:inset 0px 1px 0px 0px #bee2f9;
            box-shadow:inset 0px 1px 0px 0px #bee2f9;
            background:-webkit-gradient( linear, left top, left bottom, color-stop(0.05,
#63b8ee), color-stop(1, #468ccf) );
            background:-moz-linear-gradient( center top, #63b8ee 5%, #468ccf 100% );
            filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#63b8ee',
endColorstr='#468ccf');
            background-color:#63b8ee;
            -webkit-border-top-left-radius:0px;
            -moz-border-radius-topleft:0px;
            border-top-left-radius:0px;
            -webkit-border-top-right-radius:0px;
            -moz-border-radius-topright:0px;
            border-top-right-radius:0px;
            -webkit-border-bottom-right-radius:0px;
            -moz-border-radius-bottomright:0px;
            border-bottom-right-radius:0px;
            -webkit-border-bottom-left-radius:0px;
            -moz-border-radius-bottomleft:0px;
            border-bottom-left-radius:0px;
            text-indent:0;
            border:1px solid #3866a3;
            display:inline-block;
            color:#14396a;
            font-family:Arial;
            font-size:15px;
            font-weight:bold;
            font-style:normal;
            height:65px;
            line-height:65px;
            width:131px;
            text-decoration:none;
            text-align:center;
            text-shadow:1px 1px 0px #97b2cc;
}
.Test:hover {
            background:-webkit-gradient( linear, left top, left bottom, color-stop(0.05,
#468ccf), color-stop(1, #63b8ee) );
            background:-moz-linear-gradient( center top, #468ccf 5%, #63b8ee 100% );
            filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#468ccf',
endColorstr='#63b8ee');
            background-color:#468ccf;
}.Test:active {
            position:relative;
            top:1px;
}

.centered{
    margin: 0 auto;
    text-align: left;
    width: 800px;
}


using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
```

```
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using RateMyDebate.Models;

namespace RateMyDebate.Controllers
{
    public class CategoryController : Controller
    {
        private RateMyDebateContext db = new RateMyDebateContext();

        // GET: /Category/
        public ActionResult Index()
        {
            return View(db.Categories.ToList());
        }

        // GET: /Category/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Category category = db.Categories.Find(id);
            if (category == null)
            {
                return HttpNotFound();
            }
            return View(category);
        }

        // GET: /Category/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: /Category/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include="CategoryId,CategoryName")] Category
category)
        {
            if (ModelState.IsValid)
            {
                db.Categories.Add(category);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(category);
        }

        // GET: /Category/Edit/5
        public ActionResult Edit(int? id)
```

```
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Category category = db.Categories.Find(id);
            if (category == null)
            {
                return HttpNotFound();
            }
            return View(category);
        }

        // POST: /Category/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include="CategoryId,CategoryName")] Category
category)
        {
            if (ModelState.IsValid)
            {
                db.Entry(category).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(category);
        }

        // GET: /Category/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Category category = db.Categories.Find(id);
            if (category == null)
            {
                return HttpNotFound();
            }
            return View(category);
        }

        // POST: /Category/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Category category = db.Categories.Find(id);
            db.Categories.Remove(category);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
```

```
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}


using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Security.Cryptography;
using System.Web;
using System.Web.Mvc;
using Antlr.Runtime;
using RateMyDebate.Migrations;
using RateMyDebate.Models;
using RateMyDebate.ViewModels;

namespace RateMyDebate.Controllers
{
    public class DebateController : Controller
    {
        private RateMyDebateContext db = new RateMyDebateContext();
        private readonly IDebateRepository _debateRepository;


        public DebateController(IDebateRepository debateRepository)
        {
            _debateRepository = debateRepository;
        }

        // GET: /Debate/
        /*
        public ActionResult Index()
        {

            //return View(db.Debate.ToList());
            VM.User = db.UserInformation.ToList();
            VM.Debate = db.Debate.ToList();
            VM.Categories = db.Categories.ToList();

            return View(VM);
        }*/

        // GET: /Debate/
        public ActionResult Index(String category, String creator, String challenger)

        {
            List<Debate> myList = db.Debate.ToList().Where(x =>
x.Live.Equals(true)).ToList();
            DebateUser VM = new DebateUser();

            var CategoryQry = from d in db.Categories
                              orderby d.CategoryName
                              select d.CategoryName;
```

```
            VM.User = db.UserInformation.ToList();
            VM.Debate = db.Debate.ToList();
            VM.Categories = db.Categories.ToList();


            if (!String.IsNullOrEmpty(category))
            {
                myList = myList.Where(x =>
x.CategoryId.CategoryName.Equals(category)).ToList();
            }

            if (!String.IsNullOrEmpty(creator))
            {
                myList = myList.Where(x => x.CreatorId.nickName.Contains(creator)).ToList();
            }

            if (!String.IsNullOrEmpty(challenger))
            {
                myList = myList.ToList().Where(x =>
x.ChallengerId.nickName.Contains(challenger)).ToList();
            }

            VM.Debate = myList;

            ViewBag.category = new SelectList(CategoryQry);


            return View(VM);
        }

        // GET: /Debate/Details/5
        public ActionResult Details(int? id)
        {

            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            Debate debate = db.Debate.Find(id);

            if (debate == null)
            {
                return HttpNotFound();
            }

            return View(debate);
        }

        public ActionResult Display(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            DebateDisplayViewModel DDVM = new DebateDisplayViewModel();

            DDVM.Debate = db.Debate.Find(id);
```

```
            /*
            DDVM.CreatorInformation = db.UserInformation.ToList();
            DDVM.ChallengerInformation = db.UserInformation.ToList();
            DDVM.Category = db.Categories.ToList();
            */
            UserInformation creator = db.UserInformation.Find(DDVM.Debate.CreatorIdId);
            DDVM.CreatorInformation = creator;

            UserInformation challenger =
db.UserInformation.Find(DDVM.Debate.ChallengerIdId);
            DDVM.ChallengerInformation = challenger;

            Category category =  db.Categories.Find(DDVM.Debate.CategoryIdId);
            DDVM.Category = category;


            /*
           if (DDVM.Debate == null || DDVM.Category == null || DDVM.CreatorInformation ==
null || DDVM.ChallengerInformation == null)
            {
                return HttpNotFound();
            }*/
            return View(DDVM);
        }

        // GET: /Debate/Create
        public ActionResult Create()
        {
            var CategoryQry = from d in db.Categories
                              orderby d.CategoryName
                              select d.CategoryName;

            var CategoryQryIds = from d in db.Categories
                              orderby d.CategoryName
                              select d.CategoryId;
            ViewData["Categories"] = new SelectList(CategoryQryIds, CategoryQry);
            return View();
        }

        // POST: /Debate/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult
Create([Bind(Include="creatorIdId,DebateId,Subject,Description,CategoryIdId,Timelimit")]
Debate debate)
        {
            debate.DateTime = DateTime.Now;
            debate.Live = true;

            if (ModelState.IsValid)
            {
                _debateRepository.AddDebate(debate);
                _debateRepository.Save();
                String url = "LiveChat?id=" + debate.DebateId;
                return Redirect(url);
            }
```

```
            return View(debate);
        }

        // GET: /Debate/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Debate debate = db.Debate.Find(id);
            if (debate == null)
            {
                return HttpNotFound();
            }
            return View(debate);
        }

        // POST: /Debate/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult
Edit([Bind(Include="DebateId,Subject,Description,ChatText,CreatorVotes,ChallengerVotes,Live,
DateTime")] Debate debate)
        {
            if (ModelState.IsValid)
            {
                db.Entry(debate).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(debate);
        }

        // GET: /Debate/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Debate debate = db.Debate.Find(id);
            if (debate == null)
            {
                return HttpNotFound();
            }
            return View(debate);
        }

        // POST: /Debate/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Debate debate = db.Debate.Find(id);
            db.Debate.Remove(debate);
```

```
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        public ActionResult LiveChat(int? id)
        {
            DebateDisplayViewModel DDVM = FindDebateDisplayViewModel(id);

            ViewBag.Title = DDVM.Debate.Subject;

            return View(DDVM);
        }

        [HttpPost]
        public void EnterChallenger(int debateId)
        {
            var user = Session["UserInfoSession"] as UserInformation;
            Debate debate = _debateRepository.FindDebate(debateId);

            if (debate.ChallengerIdId == null)
            {
                debate.ChallengerIdId = user.userInformationId;
                _debateRepository.UpdateDebate(debate);
                _debateRepository.Save();
            }
        }

        [HttpPost]
        public void SaveMessage(String sender, String message, int debateId)
        {
            message = message.Replace("\n", "");
            String formattedMessage = "\n" + "[" + DateTime.Now + "] " + sender + " : " +
message;
            Debate debate = _debateRepository.FindDebate(debateId);
            debate.ChatText += formattedMessage;
            _debateRepository.UpdateDebate(debate);
            _debateRepository.Save();
        }

        public Debate FindDebate(int id)
        {
            Debate debate = _debateRepository.FindDebate(id);
            return debate;
        }

        public DebateDisplayViewModel FindDebateDisplayViewModel(int? id)
        {
            DebateDisplayViewModel DDVM = new DebateDisplayViewModel();

            DDVM.Debate = db.Debate.Find(id);
            UserInformation creator = db.UserInformation.Find(DDVM.Debate.CreatorIdId);
            DDVM.CreatorInformation = creator;

            UserInformation challenger =
db.UserInformation.Find(DDVM.Debate.ChallengerIdId);
            DDVM.ChallengerInformation = challenger;

            Category category = db.Categories.Find(DDVM.Debate.CategoryIdId);
            DDVM.Category = category;
```

```
            return DDVM;
        }

        public ActionResult TimerTest()
        {
            return View();
        }

        public DebateDisplayViewModel FillDDVM(int? id)
        {
            DebateDisplayViewModel DDVM = new DebateDisplayViewModel();

            DDVM.Debate = db.Debate.Find(id);

            UserInformation creator = db.UserInformation.Find(DDVM.Debate.CreatorIdId);
            DDVM.CreatorInformation = creator;

            UserInformation challenger =
db.UserInformation.Find(DDVM.Debate.ChallengerIdId);
            DDVM.ChallengerInformation = challenger;

            Category category = db.Categories.Find(DDVM.Debate.CategoryIdId);
            DDVM.Category = category;

            return DDVM;
        }

        public void PlaceVote(int debateId, int votePosition)
        {
            var user = Session["UserInfoSession"] as UserInformation;
            int voterId = user.userInformationId;

            Vote vote = FindVote(voterId, debateId);
            vote.VotePos = votePosition;

            if (ModelState.IsValid)
            {
                db.Entry(vote).State = EntityState.Modified;
                db.SaveChanges();
            }

        }

        public Vote FindVote(int voterId, int debateId)
        {
            Vote vote = db.Votes.Find(voterId, debateId);
            return vote;
        }

        public ActionResult ShowEndScreen(DebateDisplayViewModel ddvm)
        {

            return View();
        }

        public String ProcessDebateResult(int debateId)
        {
            Debate debate = _debateRepository.FindDebate(debateId);
            int creatorId = debate.CreatorIdId;
            int? challengerId = debate.ChallengerIdId;
```

```
            UserInformation creator = db.UserInformation.Find(creatorId);
            UserInformation challenger = db.UserInformation.Find(challengerId);

            String creatorNick = creator.nickName;
            String challengerNick = challenger.nickName;

            if (challengerId == null) return "As no challenger was found, the debate will
end without a conclusion. Sorry folks!";

            Result result = new Result();
            result.DebateId = debateId;
            String endingSentence = "Not assigned";

            List<Vote> creatorVoteCount = db.Votes.ToList().Where(x => x.DebateId ==
debateId & x.VotePos == 1).ToList();
            List<Vote> challengerVoteCount =
                db.Votes.ToList().Where(x => x.DebateId == debateId & x.VotePos ==
2).ToList();

            int creatorVotesNum = creatorVoteCount.Count();
            int challengerVotesNum = challengerVoteCount.Count();

            if (creatorVotesNum > challengerVotesNum)
            {
                result.WinnerId = creatorId;
                result.LoserId = challengerId;

                endingSentence = "Thank you for participating and spectating! The winner is
" +
                 creatorNick + " with a score of " + result.CreatorVotesCount + " to " +
                 challengerNick + "'s score of " + result.ChallengerVotesCount;
            }
            else if (creatorVotesNum < challengerVotesNum)
            {
                result.WinnerId = challengerId;
                result.LoserId = creatorId;

                endingSentence = "Thank you for participating and spectating! The winner is
" +
                 challengerNick + " with a score of " + result.ChallengerVotesCount + " to "
+
                 creatorNick + "'s score of " + result.CreatorVotesCount;
            }
            else if (creatorVotesNum == challengerVotesNum)
            {
                result.Draw = true;

                endingSentence = "Thank you for participating and spectating! Unfortunately
a winner could not be found as both participants had a vote count of " + creatorVotesNum;
            }

            result.CreatorVotesCount = creatorVotesNum;
            result.ChallengerVotesCount = challengerVotesNum;

            SaveResult(result);
            SetDebateInactive(debateId);

            return endingSentence;
        }
```

```
        public void SaveResult(Result result)
        {
            Result check = FindResult(result.DebateId);
            if (check == null)
            {
                if (ModelState.IsValid)
                {
                    db.Results.Add(result);
                    db.SaveChanges();
                }
            }
            else if (check == result)
            {
                if (ModelState.IsValid)
                {
                    db.Entry(result).State = EntityState.Unchanged;
                    db.SaveChanges();
                }
            }
            else if (check != result)
            {
                db.Entry(check).CurrentValues.SetValues(result);
                db.SaveChanges();
            }
        }

        public Result FindResult(int debateId)
        {
            Result result = db.Results.Find(debateId);
            return result;
        }

        public void SetDebateInactive(int debateId)
        {
            Debate debate = _debateRepository.FindDebate(debateId);
            debate.Live = false;
            if (ModelState.IsValid)
            {
                _debateRepository.UpdateDebate(debate);
                _debateRepository.Save();
            }
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}


using RateMyDebate.Models;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Security;

namespace RateMyDebate.Controllers
{
    public class HomeController : Controller
    {
        private RateMyDebateContext db = new RateMyDebateContext();
        public ActionResult Index()
        {
            //YO
            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";

            return View();
        }

        public ActionResult Contact()
        {
            ViewBag.Message = "Contact us";

            return View();
        }
        public ActionResult Index2()
        {
            return View();
        }

        public ActionResult RasmusIndex()
        {
            return View();
        }

        [HttpGet]
        public ActionResult Login()
        {
            return View();
        }

        [HttpPost]
        public ActionResult Login(Models.userLogonModel user)
        {


            var usermodel = db.UserModel.FirstOrDefault(u => u.userName == user.userName);
            if (ModelState.IsValid)
            {

                if(IsValid(user.userName, user.Password)){

                FormsAuthentication.SetAuthCookie(user.userName, false);
                return RedirectToAction("RasmusIndex", "Home");
                }
                  else
```

```
                {
                ModelState.AddModelError("", "Logins is wrong");
                }
            }

            return View(usermodel);
        }

        private bool IsValid(string userName, string password)
        {
            var crypto = new SimpleCrypto.PBKDF2();
            bool isValid = false;
            using (var db = new RateMyDebateContext())
            {
                var user = db.UserModel.FirstOrDefault(u => u.userName == userName);
                var userinfo = db.UserInformation.FirstOrDefault(u => u.userId ==
user.accountId);
                if(user != null){
                    if (user.Password == crypto.Compute(password, user.Salt))
                    {
                        Session["UserSession"] = user;
                        Session["UserInfoSession"] = userinfo;
                        isValid = true;
                    }
                }

            }



            return isValid;
        }
        public ActionResult LogOut(){

            Session["UserSession"] = null;
            Session["UserInfoSession"] = null;
            FormsAuthentication.SignOut();
            return RedirectToAction("RasmusIndex", "Home");
        }
    }
}

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using RateMyDebate.Models;

namespace RateMyDebate.Controllers
{
    public class InboxController : Controller
    {
        private RateMyDebateContext db = new RateMyDebateContext();

        // GET: /Inbox/
        public ActionResult Index()
```

```
        {
            var inbox = db.Inbox.Include(i => i.userInformation);
          //  var inboxtry = db
            return View(inbox.ToList());
        }

        // GET: /Inbox/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Inbox inbox = db.Inbox.Find(id);
            if (inbox == null)
            {
                return HttpNotFound();
            }
            return View(inbox);
        }

        // GET: /Inbox/Create
        public ActionResult Create()
        {
            ViewBag.userId = new SelectList(db.UserInformation, "userInformationId",
"fName");
            return View();
        }

        // POST: /Inbox/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include="inboxId,userId")] Inbox inbox)
        {
            if (ModelState.IsValid)
            {
                db.Inbox.Add(inbox);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            ViewBag.userId = new SelectList(db.UserInformation, "userInformationId",
"fName", inbox.userId);
            return View(inbox);
        }

        // GET: /Inbox/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Inbox inbox = db.Inbox.Find(id);
            if (inbox == null)
            {
                return HttpNotFound();
```

```
            }
            ViewBag.userId = new SelectList(db.UserInformation, "userInformationId",
"fName", inbox.userId);
            return View(inbox);
        }

        // POST: /Inbox/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include="inboxId,userId")] Inbox inbox)
        {
            if (ModelState.IsValid)
            {
                db.Entry(inbox).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.userId = new SelectList(db.UserInformation, "userInformationId",
"fName", inbox.userId);
            return View(inbox);
        }

        // GET: /Inbox/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Inbox inbox = db.Inbox.Find(id);
            if (inbox == null)
            {
                return HttpNotFound();
            }
            return View(inbox);
        }

        // POST: /Inbox/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Inbox inbox = db.Inbox.Find(id);
            db.Inbox.Remove(inbox);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
```

```
}


using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using RateMyDebate.Models;

namespace RateMyDebate.Controllers
{
    public class MessageController : Controller
    {
        private RateMyDebateContext db = new RateMyDebateContext();

        // GET: /Message/
        public ActionResult Index()
        {
            UserInformation infoUser = new UserInformation();

            if(Session["UserSession"] != null){
                var user = Session["UserSession"];
                UserModel userr = user as UserModel;
                var usermodel = db.UserInformation.FirstOrDefault(u => u.userId ==
userr.accountId);
                infoUser = usermodel;
                var message = db.Message.Include(m => m.inboxId).Include(m =>
m.userInformation).Where(m => m.userId == infoUser.userId).OrderByDescending(m =>
m.messageId);
                return View(message.ToList());
            }

            return View("NotLoggedIn");
        }

        // GET: /Message/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Message message = db.Message.Find(id);
            if (message == null)
            {
                return HttpNotFound();
            }
            return View(message);
        }

        // GET: /Message/Create
        public ActionResult Create()
        {


            return View();
```

```
        }

        // POST: /Message/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create( Message message)
        {

            if (Session["UserSession"] != null)
            {
                var user = Session["UserSession"] as UserModel;
                message.Sender = user.userName;
                if(message.Receiver != null){

                  var usermodel = db.UserModel.FirstOrDefault(u => u.userName ==
message.Receiver);
                  if(usermodel != null)
                  {
                    var receiverUser = db.UserInformation.FirstOrDefault(u => u.userId ==
usermodel.accountId);
                    message.userId = receiverUser.userInformationId;
                    message.inboxKey = 1;
                  }
                  else
                  {
                      message = null;
                      ModelState.AddModelError("Receiver","Receiver Doesn't exist in user
database! Make sure the name is spelled correct!");
                  }

                }

            }
            if (ModelState.IsValid)
            {
                db.Message.Add(message);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(message);
        }

        // GET: /Message/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Message message = db.Message.Find(id);
            if (message == null)
            {
                return HttpNotFound();
            }
            ViewBag.inboxKey = new SelectList(db.Inbox, "inboxId", "inboxId",
message.inboxKey);
```

```
            ViewBag.userId = new SelectList(db.UserInformation, "userInformationId",
"fName", message.userId);
            return View(message);
        }

        // POST: /Message/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult
Edit([Bind(Include="messageId,userId,inboxKey,subject,messageText")] Message message)
        {
            if (ModelState.IsValid)
            {
                db.Entry(message).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            ViewBag.inboxKey = new SelectList(db.Inbox, "inboxId", "inboxId",
message.inboxKey);
            ViewBag.userId = new SelectList(db.UserInformation, "userInformationId",
"fName", message.userId);
            return View(message);
        }

        // GET: /Message/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Message message = db.Message.Find(id);
            if (message == null)
            {
                return HttpNotFound();
            }
            return View(message);
        }

        // POST: /Message/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Message message = db.Message.Find(id);
            db.Message.Remove(message);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
```

```
        }
    }
}


using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using RateMyDebate.Models;
using RateMyDebate.ViewModels;
using System.Web.UI;

namespace RateMyDebate.Controllers
{
    [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
    public class UserController : Controller
    {
        private RateMyDebateContext db = new RateMyDebateContext();
        private UserCreateViewModel vm = new UserCreateViewModel();
        private UserModel newUserModel = new UserModel();
        // GET: /User/
        public ActionResult Index()
        {

            return View(db.UserModel.ToList());
        }

        // GET: /User/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            UserModel usermodel = db.UserModel.Find(id);

            UserModel userSession = Session["userSession"] as UserModel;
            if (usermodel == null)
            {
                return HttpNotFound();
            }
            return View(usermodel);
        }

        // GET: /User/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: /User/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
```

```
        [ValidateAntiForgeryToken]
        public ActionResult Create(UserModel usermodel)
        {


            if (ModelState.IsValid)
            {
                //var id = usermodel;

               // db.UserModel.Add(usermodel);
               //db.SaveChanges();

                TempData["Id"] = usermodel;

                return RedirectToAction("Create", "UserInformation");
            }

            return View(usermodel);
        }

        // GET: /User/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            UserModel usermodel = db.UserModel.Find(id);
            if (usermodel == null)
            {
                return HttpNotFound();
            }
            return View(usermodel);
        }

        // POST: /User/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include="accountId,userName,Password")] UserModel
usermodel)
        {
            if (ModelState.IsValid)
            {
                db.Entry(usermodel).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(usermodel);
        }

        // GET: /User/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
```

```
            UserModel usermodel = db.UserModel.Find(id);
            if (usermodel == null)
            {
                return HttpNotFound();
            }
            return View(usermodel);
        }

        // POST: /User/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            UserModel usermodel = db.UserModel.Find(id);
            db.UserModel.Remove(usermodel);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }

        [HttpPost]
        public JsonResult ValidateUserName(string userName)
        {
           // !db.UserModel.Any(user => user.userName == userName

            //!db.UserModel.Any(user => user.userName.Equals(userName)

            var usercheck = db.UserModel.FirstOrDefault(user =>
user.userName.Equals(userName));

            return Json(usercheck == null);

        }


    }
}


using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web;
using System.Web.Mvc;
using RateMyDebate.Models;
using System.Web.UI;
using System.Collections;

namespace RateMyDebate.Controllers
```

```
{
    [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
    public class UserInformationController : Controller
    {


        private RateMyDebateContext db = new RateMyDebateContext();

        // GET: /UserInformation/
        public ActionResult Index()
        {
            return View(db.UserInformation.ToList());
        }

        // GET: /UserInformation/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }

            UserInformation userinformation = db.UserInformation.Find(id);
            int i = userinformation.userId;
          //  var user = from m in db.UserInformation where m.userInformationId == id select
m;

           //var userTry = from m in db.UserModel where
m.accountId.Equals(userinformation.accountId.accountId) select m;

            //userinformation.accountId = db.UserModel.Find(36);


            if (userinformation == null)
            {
                return HttpNotFound();
            }
            return View(userinformation);
        }

        // GET: /UserInformation/Create
        public ActionResult Create()
        {
            return View();
        }

        // POST: /UserInformation/Create
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult
Create([Bind(Include="userInformationId,fName,lName,nickName,age,autobiography,Email")]
UserInformation userinformation)
        {
            if (ModelState.IsValid)
            {
                var id = TempData["Id"] as UserModel;
                var crypto = new SimpleCrypto.PBKDF2();
```

```
                var encryptPass = crypto.Compute(id.Password);
                id.Password = encryptPass;
                id.ConfirmPassword = crypto.Compute(id.ConfirmPassword);
                id.Salt = crypto.Salt;
                userinformation.accountId = id;

                db.UserInformation.Add(userinformation);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return RedirectToAction("Index");
        }

        // GET: /UserInformation/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            if (Session["UserInfoSession"] != null)
            {
                var session = Session["UserinfoSession"] as UserInformation;
                if(session.userInformationId == id){
                    UserInformation userinformation = db.UserInformation.Find(id);
                    userinformation.accountId = db.UserModel.Find(userinformation.userId);
                    if (userinformation == null)
                    {
                        return HttpNotFound();
                    }
                    return View(userinformation);
                }
            }
            ViewBag.Error = "Error!";
            return View("UserEditError");
        }

        // POST: /UserInformation/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit( UserInformation userinformation)
        {

            if (ModelState.IsValid)
            {
                db.Entry(userinformation).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(userinformation);
        }

        // GET: /UserInformation/Delete/5
        public ActionResult Delete(int? id)
        {
```

```
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    if (Session["UserInfoSession"] != null)
    {
        var session = Session["UserinfoSession"] as UserInformation;
        if (session.userInformationId == id)
        {
            UserInformation userinformation = db.UserInformation.Find(id);
            userinformation.accountId = db.UserModel.Find(userinformation.userId);
            if (userinformation == null)
            {
                return HttpNotFound();
            }
            return View(userinformation);
        }
    }
    return View("UserEditError");
}

// POST: /UserInformation/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public ActionResult DeleteConfirmed(int id)
{
    UserInformation userinformation = db.UserInformation.Find(id);
    db.UserInformation.Remove(userinformation);
    db.SaveChanges();
    return RedirectToAction("Index");
}

protected override void Dispose(bool disposing)
{
    if (disposing)
    {
        db.Dispose();
    }
    base.Dispose(disposing);
}
public ActionResult profile()
{
    UserInformation user = new UserInformation();
    UserModel sessionUser = Session["userSession"] as UserModel;
    int id = sessionUser.accountId;

    var user1 = db.UserInformation.FirstOrDefault(u => u.userId == id);

    user = user1;

    return View(user);
}

public ActionResult Profiles(String SearchName)
{
    List<UserInformation> user;
    user = db.UserInformation.ToList();
   // ViewBag.NotFound = "";
    if(SearchName != null){
```

```
                user = db.UserInformation.Where(u => u.nickName == SearchName).ToList();
                if (user.Count() == 0)
                {
                    ViewBag.NotFound = "User not found! Check your spelling!";
                    return View(user);
                }
            }
            return View(user);
        }


        public ActionResult Edit2(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            if (Session["UserSession"] != null)
            {
                var session = Session["UserSession"] as UserModel;
                if (session.accountId == id)
                {
                    UserModel usermodel = db.UserModel.Find(id);
                   // usermodel.accountId = db.UserModel.Find(userinformation.userId);
                    if (usermodel == null)
                    {
                        return HttpNotFound();
                    }
                    return View(usermodel);
                }
            }
            return View("UserEditError");
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit2(UserModel usermodel)
        {
            if (ModelState.IsValid)
            {
                var crypto = new SimpleCrypto.PBKDF2();
                usermodel.Password = crypto.Compute(usermodel.Password);
                usermodel.ConfirmPassword = crypto.Compute(usermodel.ConfirmPassword);
                usermodel.Salt = crypto.Salt;
                db.Entry(usermodel).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("profile");
            }
            return View(usermodel);
        }

        [HttpPost]
        public JsonResult ValidateNickName(string nickName)
        {
            // !db.UserModel.Any(user => user.userName == userName

            //!db.UserModel.Any(user => user.userName.Equals(userName)

            var usercheck = db.UserInformation.FirstOrDefault(user =>
user.nickName.Equals(nickName));
```

```
            return Json(usercheck == null);

        }


    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Threading;
using System.Threading.Tasks;
using System.Web;
using System.Web.Hosting;
using Microsoft.AspNet.SignalR;
using Microsoft.AspNet.SignalR.Hubs;
using RateMyDebate.Models;

namespace RateMyDebate.Hubs
{

    public class ChatHub : Hub
    {
        static readonly HashSet<int> DebateGroups = new HashSet<int>();
        private Timer _timer;
        private int time;

        public void JoinDebateGroup(int debateId)
        {
            String Id = debateId.ToString();
            if (!DebateGroups.Contains(debateId))
            {
                CreateDebateGroup(debateId);
            }
            else
            {
                Groups.Add(Context.ConnectionId, Id);
            }
        }

        public void CreateDebateGroup(int debateId)
        {
            String Id = debateId.ToString();
            if(DebateGroups.Contains(debateId)) return;
            DebateGroups.Add(debateId);
            JoinDebateGroup(debateId);
            Clients.All.debateGroups(new[] {debateId});
        }

        public void StartTimer(int debateId)
        {
            time = 0;
            _timer = new Timer(UpdateTimer, debateId, TimeSpan.FromSeconds(0),
TimeSpan.FromSeconds(1));
        }

        private void UpdateTimer(Object state)
```

```
        {
            time += 1;
            BroadcastTimer(state);
        }

        private void BroadcastTimer(Object state)
        {
            String debateId = state.ToString();

            Clients.Group(debateId).broadcastTime(time);
        }

        public void Send(string name, string message, int debateId)
        {
            String debate = debateId.ToString();

            Clients.Group(debate).broadcastMessage(name, message);
        }
    }

}



using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class Category
    {
        [Key]
        public int CategoryId { get; set; }

        [Display(Name = "Category")]
        [StringLength(30, ErrorMessage = "Not allowed to use more than 30 characters in a
category name")]
        public String CategoryName { get; set; }
    }
}

using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace RateMyDebate.Models
{
    public class Debate
    {
        public int CreatorIdId { get; set; }
        public int? ChallengerIdId { get; set; }
        public int CategoryIdId { get; set; }
```

```
        [Key]
        public int DebateId { get; set; }
        [Display(Name = "Topic")]
        public String Subject { get; set; }
        [Display (Name = "Case")]
        public String Description { get; set; }
        public String ChatText { get; set; }

        [ForeignKey("CreatorIdId")]
        public UserInformation CreatorId { get; set; }

        [ForeignKey("ChallengerIdId")]
        public UserInformation ChallengerId { get; set; }

        [ForeignKey("CategoryIdId")]
        public Category CategoryId { get; set; }

        public Boolean Live { get; set; }
        [Display (Name = "Created")]
        public DateTime DateTime { get; set; }

        public int TimeLimit { get; set; }
        public IEnumerator GetEnumerator()
        {
            throw new NotImplementedException();
        }


    }
}

using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class DebateRepository : IDebateRepository
    {
        RateMyDebateContext db = new RateMyDebateContext();

        public void Dispose()
        {
            db.Dispose();
        }

        public IQueryable<Debate> GetDebates
        {
            get { return db.Debate; }
        }
        public Debate FindDebate(int? debateId)
        {
            Debate debate = db.Debate.Find(debateId);
            return debate;
        }

        public void AddDebate(Debate debate)
```

```
        {
            db.Debate.Add(debate);
        }

        public void UpdateDebate(Debate debate)
        {
            db.Entry(debate).State = EntityState.Modified;
        }

        public void DeleteDebate(int debateId)
        {
            var debate = db.Debate.Find(debateId);
            db.Debate.Remove(debate);
        }

        public void Save()
        {
            db.SaveChanges();
        }
    }
    public interface IDebateRepository : IDisposable
    {
        IQueryable<Debate> GetDebates { get; }
        Debate FindDebate(int? debateId);
        void AddDebate(Debate debate);
        void UpdateDebate(Debate debate);
        void DeleteDebate(int debateId);
        void Save();
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class Friendship
    {
        [Key]
        public int FriendshipId { get; set; }
        public int FriendOneId { get; set; }
        public int FriendTwoId { get; set; }

        [ForeignKey("FriendOneId")]
        public UserInformation FriendOne { get; set; }
        [ForeignKey("FriendTwoId")]
        public UserInformation FriendTwo { get; set; }

        public DateTime Since { get; set; }
        public Boolean Accepted { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
```

```
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class Inbox
    {
        [Key]
        public int inboxId { get; set; }

        [ForeignKey("userInformation")]
        public int userId { get; set; }
        public virtual UserInformation userInformation { get; set; }


        [ForeignKey("messageId")]
        public ICollection<Message> messageId { get; set; }


    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class Message
    {
        [Key]
        public int messageId { get; set; }

        [ForeignKey("userInformation")]
        public int userId { get; set; }
        public virtual UserInformation userInformation { get; set; }

        [ForeignKey("inboxId")]
        public int inboxKey { get; set; }

        public Inbox inboxId { get; set; }


        public String Sender { get; set; }

        [Required]
        public String Receiver { get; set; }
        [Required]
        public String  subject  { get; set; }
        [Required]
        public String messageText { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class MessageMap
    {
        public int InboxId { get; set; }


        [Key, Column(Order = 1)]
        public int MessageId { get; set; }

        [Key, Column(Order = 2)]
        public int UserId { get; set; }

        [ForeignKey("UserId")]
        public UserInformation User { get; set; }

        [ForeignKey("MessageId")]
        public Message Message { get; set; }

        [ForeignKey("InboxId")]
        public Inbox Inbox { get; set; }

        public Boolean IsRead { get; set; }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.Entity.ModelConfiguration.Conventions;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class RateMyDebateContext :DbContext
    {
        public DbSet<UserModel> UserModel { get; set; }
        public DbSet<UserInformation> UserInformation { get; set; }

        public DbSet<Inbox> Inbox { get; set; }

        public DbSet<Message> Message { get; set; }
        public DbSet<Debate> Debate { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Vote> Votes { get; set; }
        public DbSet<MessageMap> MessageMaps { get; set; }

        public DbSet<Result> Results { get; set; }
        public DbSet<Friendship> Friendships { get; set; }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();
```

```
            modelBuilder.Entity<UserInformation>()
            .HasRequired(u => u.accountId)
            .WithMany()
            .WillCascadeOnDelete(false);
        }
    }


}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class Result
    {


        public int? WinnerId { get; set; }
        public int? LoserId { get; set; }


        public Debate Debate { get; set; }

        [Key, ForeignKey("Debate")]
        public int DebateId { get; set; }

        [ForeignKey("WinnerId")]
        public UserInformation Winner { get; set; }

        [ForeignKey("LoserId")]
        public UserInformation Loser { get; set; }

        public Boolean? Draw { get; set; }
        public int? CreatorVotesCount { get; set; }
        public int? ChallengerVotesCount { get; set; }
    }
}
```

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

namespace RateMyDebate.Models
{
    public class UserInformation
    {
        [Key]
        public int userInformationId { get; set; }
```

```
        [Required]
        [Display(Name = "First name")]
        public String fName { get; set; }

        [Display(Name = "Last name")]
        [Required]
        public String lName { get; set; }

        [Required]
        [Display(Name = "Username")]
        [Remote("ValidateNickName", "UserInformation", HttpMethod = "POST", ErrorMessage =
"User name already exists. Please enter a different user name.")]
        public String nickName { get; set; }

        [Display(Name = "Age")]
        [Required]
        public int age { get; set; }

        [Display(Name = "Autobiography")]
        [Required]
        public String autobiography { get; set; }

        [Display(Name = "Email")]
        [Required]
        public String Email { get; set; }

        [ForeignKey("accountId")]
        public int userId { get; set; }
        public UserModel accountId { get; set; }
    }
}


using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace RateMyDebate.Models
{
    public class userLogonModel
    {


        [Display(Name = "User name")]
        [Required]
        [Remote("ValidateUserName", "User", HttpMethod = "POST", ErrorMessage = "User name
already exists. Please enter a different user name.")]
        public String userName { get; set; }

        [Display(Name = "Password")]
        [Required]
        public String Password { get; set; }

        public String Salt { get; set; }
    }
```

```
}
```

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

namespace RateMyDebate.Models
{
    public class UserModel
    {
        [Key]
        public int accountId { get; set; }


        [Display(Name = "User name")]
        [Required]
        [Remote("ValidateUserName", "User", HttpMethod= "POST", ErrorMessage = "User name
already exists. Please enter a different user name.")]
        public String userName { get; set; }

        [Display(Name = "Password")]
        [Required]
        public String Password { get; set; }

        public String Salt { get; set; }

        [NotMapped]
        [System.ComponentModel.DataAnnotations.Compare("Password", ErrorMessage = "Password
does not match!")]
        public String ConfirmPassword { get; set; }

        public ICollection<UserInformation> UserInformation { get; set; }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class Vote
    {

        [Key, Column(Order = 0)]
        public int VoterId { get; set; }

        [Key, Column(Order = 1)]
        public int DebateId { get; set; }

        [ForeignKey("VoterId")]
        public UserInformation Voter { get; set; }
```

```
        [ForeignKey("DebateId")]
        public Debate Debate { get; set; }
        public int VotePos { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Web;

namespace RateMyDebate.Models
{
    public class VoteList
    {
        public int VoterId { get; set; }
        public int DebateId { get; set; }

        [Key]
        public int VoteListId { get; set; }

        [ForeignKey("VoterId")]
        public UserInformation Voter { get; set; }

        [ForeignKey("DebateId")]
        public Debate Debate { get; set; }
        public int Vote { get; set; }

    }
}

using System.Collections.Generic;
using RateMyDebate.Models;

namespace RateMyDebate.ViewModels
{
    public class DebateDisplayViewModel
    {
        public Debate Debate { get; set; }
        public Category Category { get; set; }
        public UserInformation CreatorInformation { get; set; }
        public UserInformation ChallengerInformation { get; set; }

    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using RateMyDebate.Models;

namespace RateMyDebate.ViewModels
{
    public class DebateUser
    {
```

```
        public List<UserInformation> User { get; set; }
        public List<Debate> Debate { get; set; }

        public List<Category> Categories { get; set; }


    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using RateMyDebate.Models;

namespace RateMyDebate.ViewModels
{
    public class UserCreateViewModel
    {
        public UserModel User { get; set; }
        public UserInformation UserInformation { get; set; }
    }
}
```

```
@model RateMyDebate.Models.Category

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Category</h4>
        <hr />
        @Html.ValidationSummary(true)

        <div class="form-group">
            @Html.LabelFor(model => model.CategoryName, new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.CategoryName)
                @Html.ValidationMessageFor(model => model.CategoryName)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
```

```
</div>


@model RateMyDebate.Models.Category

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Category</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.CategoryName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.CategoryName)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>


@model RateMyDebate.Models.Category

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Category</h4>
            <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.CategoryName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.CategoryName)
        </dd>

    </dl>
</div>
<p>
```

```
    @Html.ActionLink("Edit", "Edit", new { id = Model.CategoryId }) |
    @Html.ActionLink("Back to List", "Index")
</p>


@model RateMyDebate.Models.Category

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Category</h4>
        <hr />
        @Html.ValidationSummary(true)
        @Html.HiddenFor(model => model.CategoryId)

        <div class="form-group">
            @Html.LabelFor(model => model.CategoryName, new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.CategoryName)
                @Html.ValidationMessageFor(model => model.CategoryName)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@model IEnumerable<RateMyDebate.Models.Category>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.CategoryName)
```

```
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.CategoryName)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.CategoryId }) |
            @Html.ActionLink("Details", "Details", new { id=item.CategoryId }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.CategoryId })
        </td>
    </tr>
}

</table>



@using System.Web.UI.WebControls
@using Microsoft.Ajax.Utilities
@using RateMyDebate.Models
@using RateMyDebate.ViewModels
@model RateMyDebate.ViewModels.DebateDisplayViewModel

@{
    ViewBag.Title = "LiveChat";
    @Styles.Render("~/Content/css")
    @Styles.Render("~/Content/RMD.css")
    @Scripts.Render("~/bundles/modernizr")
    var user = Session["UserInfoSession"] as UserInformation;
}


<div class="messageDiv">
    <input type="button" id="sendmessage" value="Send" class="chatButton" />
    <input type="text" id="message" class="chatTextBox" />
    <input type="button" id="startTimer" value="Start!" class="chatButton" />
</div>

<div></div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script src="~/Scripts/jquery.signalR-2.1.1.min.js"></script>
<script src="~/Scripts/Timer.js"></script>

<script src="/signalr/hubs"></script>
<script type="text/javascript">
    $(function() {
        var chat = $.connection.chatHub;
        var endingTimer;
        var client = chat.client;
        var server = chat.server;
        var debateId = '@Model.Debate.DebateId';
        // $('.timerDiv').stopwatch().stopwatch('start');
```

```
        $('#message').focus();

        $.connection.hub.start().done(function () {
            server.joinDebateGroup(debateId);

            $('#sendmessage').click(function () {

                var sender = '@user.nickName';
                var message = $('#message').val();

                chat.server.send(sender, $('#message').val(), debateId);
                $('#message').val('').focus();


                $.ajax({
                    url: "/Debate/SaveMessage",
                    type: "POST",
                    data: {
                        sender: sender,
                        message: message,
                        debateId: debateId
                    }
                });
            });


            $('#startTimer').click(function() {
                server.startTimer(debateId);
            });
        });
    });

</script>


@using System.Web.UI.WebControls
@using Microsoft.Ajax.Utilities
@using RateMyDebate.Models
@using RateMyDebate.ViewModels
@model RateMyDebate.ViewModels.DebateDisplayViewModel

@{
    ViewBag.Title = "LiveChat";
    @Styles.Render("~/Content/css")
    @Styles.Render("~/Content/RMD.css")
    @Scripts.Render("~/bundles/modernizr")
    var user = Session["UserInfoSession"] as UserInformation;
}


<div class="messageDiv">
    <input type="button" id="sendmessage" value="Send" class="chatButton" />
    <input type="text" id="message" class="chatTextBox" />
</div>

<div></div>
```

```
<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script src="~/Scripts/jquery.signalR-2.1.1.min.js"></script>
<script src="~/Scripts/Timer.js"></script>

<script src="/signalr/hubs"></script>
<script type="text/javascript">
    $(function() {
        var chat = $.connection.chatHub;
        var endingTimer;
        var client = chat.client;
        var server = chat.server;
        var debateId = '@Model.Debate.DebateId';
        // $('.timerDiv').stopwatch().stopwatch('start');


        chat.client.broadcastMessage = function(name, message) {
            var formattedName;
            var formattedMessage;


            formattedName = '<div /> ' + name;
            formattedMessage = $('<div />').text(message).html();

            $('#discussion').append(formattedName + ' :' + formattedMessage);
        };

        chat.client.broadcastTime = function(time) {
            var formattedTime;
            var timeLimit = '@Model.Debate.TimeLimit';

            formattedTime = $('<div />').text(time).html() + '/' + timeLimit + ' seconds';

            $('.timerDiv').text(formattedTime).html();

            if (time == timeLimit) {
                $('.descriptionDiv').append('DONEDONEDONEDONEDONDENOEONONDENODE');

            }
        };


        $('#message').focus();

        $.connection.hub.start().done(function () {
            server.joinDebateGroup(debateId);

            $('#sendmessage').click(function () {

                var sender = '@user.nickName';
                var message = $('#message').val();

                chat.server.send(sender, $('#message').val(), debateId);
                $('#message').val('').focus();


                $.ajax({
                    url: "/Debate/SaveMessage",
                    type: "POST",
```

```
                    data: {
                        sender: sender,
                        message: message,
                        debateId: debateId
                    }
                });
            });
        });
    });

</script>


@using System.Web.Mvc.Html
@using RateMyDebate.Models
@model RateMyDebate.Models.Debate

@{
    ViewBag.Title = "Create";
    var user = Session["UserInfoSession"] as UserInformation;
}

@if (user == null)
{
    <div>You must be logged in to create a debate.</div>
}
else
{
<h2>Create</h2>

using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Debate</h4>
        <hr />
        @Html.ValidationSummary(true)

        <div class="form-group">
            @Html.LabelFor(model => model.Subject, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.Subject)
                @Html.ValidationMessageFor(model => model.Subject)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Description, new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Description)
                @Html.ValidationMessageFor(model => model.Description)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.CategoryId.CategoryName, new { @class = "control-
label col-md-2" })
```

```
            @Html.DropDownListFor(m => m.CategoryIdId,
(IEnumerable<SelectListItem>)ViewData["Categories"])
        </div>



        <div class="form-group">
            @Html.LabelFor(model => model.TimeLimit, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.TimeLimit)
                @Html.ValidationMessageFor(model => model.TimeLimit)
            </div>
        </div>

        @Html.Hidden("creatorIdId", user.userInformationId);

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>


}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

}

<script src="~/Scripts/jquery-1.10.2.js"></script>
<script type="text/javascript">
</script>

@model RateMyDebate.Models.Debate

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Debate</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Subject)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Subject)
        </dd>

        <dt>
```

```
            @Html.DisplayNameFor(model => model.Description)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Description)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.ChatText)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.ChatText)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.CreatorVotes)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.CreatorVotes)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.ChallengerVotes)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.ChallengerVotes)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Live)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Live)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.DateTime)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateTime)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

```
@model RateMyDebate.Models.Debate

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Debate</h4>
            <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.Subject)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Subject)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Description)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Description)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.ChatText)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.ChatText)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.CreatorVotes)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.CreatorVotes)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.ChallengerVotes)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.ChallengerVotes)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Live)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Live)
        </dd>
```

```
        <dt>
            @Html.DisplayNameFor(model => model.DateTime)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.DateTime)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.DebateId }) |
    @Html.ActionLink("Back to List", "Index")
</p>


@model RateMyDebate.ViewModels.DebateDisplayViewModel

@{
    ViewBag.Title = "Display";
}

<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<h2>Display archive page for debate #@Html.DisplayFor(m =>
m.Debate.CategoryId.CategoryId)</h2>

<div>
    <div class="displayDiv">
        <p style="font-weight: bold; display: inline;">@Html.DisplayNameFor(m =>
m.Debate.CategoryId.CategoryName):</p> @Html.DisplayFor(m =>
m.Debate.CategoryId.CategoryName)<br />
        <p style="font-weight: bold; display: inline;"> @Html.DisplayNameFor(m =>
m.Debate.Subject): </p> @Html.DisplayFor(m => m.Debate.Subject)

        <br /><br />
        <p style="font-weight: bold; display: inline;">Creator:</p> @Html.DisplayFor(m =>
m.Debate.CreatorId.nickName)  DISPLAY votes <br/>
        <p style="font-weight: bold; display: inline;">Challenger:</p> @Html.DisplayFor(m =>
m.Debate.ChallengerId.nickName) DISPLAY votes)

    </div>
    <div></div>
    <table class="displayTable">
        <tr>
            <td>
                @Html.DisplayNameFor(m => m.Debate.Description): @Html.DisplayFor(m =>
m.Debate.Description)
            </td>
        </tr>
        <tr>
            <td>
                @Html.DisplayFor(m => m.Debate.ChatText)
            </td>
        </tr>
    </table>
</div>


<br />
Creator: @Html.DisplayFor(m => m.Debate.CreatorId.nickName)
```

```
@Html.DisplayNameFor(m => m.Debate.Description)

@Html.DisplayFor(m => m.Debate.Description)
```

```
@model RateMyDebate.Models.Debate

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Debate</h4>
        <hr />
        @Html.ValidationSummary(true)
        @Html.HiddenFor(model => model.DebateId)

        <div class="form-group">
            @Html.LabelFor(model => model.Subject, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.Subject)
                @Html.ValidationMessageFor(model => model.Subject)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Description, new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Description)
                @Html.ValidationMessageFor(model => model.Description)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ChatText, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.ChatText)
                @Html.ValidationMessageFor(model => model.ChatText)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Live, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Live)
                @Html.ValidationMessageFor(model => model.Live)
            </div>
```

```
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.DateTime, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.DateTime)
                @Html.ValidationMessageFor(model => model.DateTime)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@using Microsoft.Ajax.Utilities
@using RateMyDebate.Models
@model RateMyDebate.ViewModels.DebateUser

@{
    ViewBag.Title = "Index";
    var user = Session["UserInfoSession"] as UserInformation;
}
<script src="~/Scripts/jquery-1.10.2.min.js"></script>

@Html.Partial("_SearchPartial")


<p>
    @Html.ActionLink("Create New", "Create")
</p>
    <table class="listingTable">
    <tr>
    <th>
        @Html.DisplayNameFor(model => model.Debate.First().Subject)
    </th>
     <th>
        @Html.DisplayNameFor(model => model.Debate.First().CategoryId.CategoryName)
    </th>
    <th>
        Creator
    </th>
    <th>
        DISPLAY CREATOR VOTES
    </th>
    <th>
        Challenger
    </th>
    <th>
        DISPLAY CHALLENGER VOTES
    </th>
```

```
    <th>
        @Html.DisplayNameFor(model => model.Debate.First().DateTime)
    </th>
    </tr>

@foreach (var item in Model.Debate)
{

    <tr>
        <td data-id="@Html.Raw(item.DebateId)" class="DisplayPage">
            @Html.DisplayFor(modelItem => item.Subject)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CategoryId.CategoryName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.CreatorId.nickName)
        </td>
        <td>
            DISPLAY CREATOR VOTES
        </td>
        @if (user == null & item.ChallengerId==null)
            {
                <td>
                    No challenger yet
                </td>
            }
            else if (item.ChallengerId != null)
            {
                 <td>
                    @Html.DisplayFor(modelItem => item.ChallengerId.nickName)
                </td>
            }
        else if (item.ChallengerId == null & user.userInformationId != item.CreatorIdId)
        {
            <td id="joinChallenger" data-id="@Html.Raw(item.DebateId)">
                No challenger yet. Join?
            </td>
        }
        else if (item.ChallengerId == null & user.userInformationId == item.CreatorIdId)
        {
            <td>
                You still have no challenger
            </td>
            }
        <td>
            DISPLAY CHALLENGER VOTES
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.DateTime)
        </td>
    </tr>
}

</table>


<script type="text/javascript">
    $(document).ready(function () {
```

```
        $("tr:even").css("border-top", "1px solid rgba(167, 166, 255, 0.35)");
        $("tr:even").css("border-bottom", "1px solid rgba(167, 166, 255, 0.35)");
        $("tr:even").css("background-color", "rgba(167, 166, 255, 0.25)");
        $("tr:odd").css("background-color", "rgba(167, 166, 255, 0.15)");
        $("tr:hover").css("background-color", "rgba(167, 166, 255, 0.75)");

    });

    $(document).ready(
        function () {
        $('.DisplayPage').on("click", function () {
            var id = $(this).attr('data-id');
            window.location = "LiveChat?id=" + id;
        });


        });

    $(document).ready(
        function() {
            $('#joinChallenger').on("click", function() {
                var id = $(this).attr('data-id');

                        $.ajax({
                          url: '/Debate/EnterChallenger',
                          type: 'Post',
                          data: { debateId: id }
                        });
                // Anvender redirect fra JavaScript, da der ikke kan performes et serverside
redirect fra et AJAX response.
                window.location = "LiveChat?id=" + id;
            });
        }
    );
</script>


@using System.Diagnostics
@using System.Text
@using System.Web.UI.WebControls
@using Microsoft.Ajax.Utilities
@using RateMyDebate.Models
@using RateMyDebate.ViewModels
@model RateMyDebate.ViewModels.DebateDisplayViewModel


@{
    ViewBag.Title = "LiveChat";
    Layout = "";
    @Styles.Render("~/Content/css")
    @Styles.Render("~/Content/RMD.css")
    @Scripts.Render("~/bundles/modernizr")
    var user = Session["UserInfoSession"] as UserInformation;

}

<div class="creatorVotesDiv">
    <div class="voteDiv" id="voteCreator">Creator</div>
    <div class="nickNameDiv" id="creatorName">@Html.DisplayFor(model =>
model.CreatorInformation.nickName)</div>
```

```
</div>

<div class="challengerVotesDiv">
    <div class="voteDiv" id="voteChallenger">
        Challenger
    </div>
    <div class="nickNameDiv" id="challengerName">@Html.DisplayFor(model =>
model.ChallengerInformation.nickName)</div>
</div>

<div class="timerDiv">0/@Model.Debate.TimeLimit seconds</div>

<div class="descriptionDiv">
    @Html.DisplayNameFor(m => m.Debate.Subject): @Html.DisplayFor(m => m.Debate.Subject)
    <br />
    @Html.DisplayNameFor(m => m.Debate.Description): @Html.DisplayFor(m =>
m.Debate.Description)
    <br />
    @Model.Debate.TimeLimit
</div>

<div class="chatDiv">
    @Html.Raw(Html.Encode(Model.Debate.ChatText).Replace("\n", "<br />"))
    <ul id="discussion"></ul>
</div>

@if (user == null)
{
    @Html.Partial("NonUser")
}
else if (user.userInformationId == Model.CreatorInformation.userInformationId)
{
    @Html.Partial("_Creator")
}
else if (user.userInformationId == Model.ChallengerInformation.userInformationId)
{
    @Html.Partial("_Participant")
}
else
{
    @Html.Partial("Spectator")
}


<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/jquery.signalR-2.1.1.min.js"></script>
<script src="~/Scripts/Timer.js"></script>
<script src="/signalr/hubs"></script>
<script type="text/javascript">

        $(function() {
            var chat = $.connection.chatHub;
            var client = chat.client;
            var server = chat.server;
            var debateConnectionId = '@Model.Debate.DebateId';

            $.connection.hub.start().done(function() {
                server.joinDebateGroup(debateConnectionId);
            });
```

```
            chat.client.broadcastMessage = function (name, message) {
                var formattedName;
                var formattedMessage;


                formattedName = '<div /> ' + name;
                formattedMessage = $('<div />').text(message).html();

                $('#discussion').append(formattedName + ' :' + formattedMessage);
            };

            chat.client.broadcastTime = function (time) {
                var formattedTime;
                var timeLimit = '@Model.Debate.TimeLimit';

                formattedTime = $('<div />').text(time).html() + '/' + timeLimit + '
seconds';

                $('.timerDiv').text(formattedTime).html();

                if (time == timeLimit) {
                    $('.timerDiv').text('Time is up!');
                    $('.chatDiv').append('<br /> The debate has come to an end! Please wait
while results are processed...');

                    var debateId = '@Model.Debate.DebateId';

                    $.ajax({
                        url: "/Debate/ProcessDebateResult",
                        type: "POST",
                        async: false,
                        datatype: "json",
                        data: {
                            debateId: debateId
                        },
                        success: function (data) {
                            $('.chatDiv').append('<br />' + data);
                        }

                    });

                }
            };
        });

</script>


@using System.Web.UI.WebControls
@using Microsoft.Ajax.Utilities
@using RateMyDebate.Models
@using RateMyDebate.ViewModels
@model RateMyDebate.ViewModels.DebateDisplayViewModel

@{
    ViewBag.Title = "LiveChat";
    @Styles.Render("~/Content/css")
    @Styles.Render("~/Content/RMD.css")
    @Scripts.Render("~/bundles/modernizr")
    var user = Session["UserInfoSession"] as UserInformation;
```

```
}

<div id="spectatorInformation">Welcome! You are a spectator. Click the creator or challenger
button to vote for your favorite debator!</div>
<div id="spectatorChat">This is the spectator chat. It is visible only to logged in
spectators. Share your thoughts with fellow spectators!</div>

<script src="~/Scripts/jquery-1.10.2.min.js"></script>

<script src="~/Scripts/jquery.signalR-2.1.1.min.js"></script>
<script src="~/Scripts/Timer.js"></script>

<script src="/signalr/hubs"></script>
<script type="text/javascript">

        $(function() {
            var chat = $.connection.chatHub;
            $.connection.hub.start();

            chat.client.broadcastTime = function (time) {
                var formattedTime;
                var timeLimit = '@Model.Debate.TimeLimit';

                formattedTime = $('<div />').text(time).html() + '/' + timeLimit + '
seconds';

                $('.timerDiv').text(formattedTime).html();

                if (time == timeLimit) {
                    $('.descriptionDiv').append('DONEDONEDONEDONEDONDENOEONONDENODE');

                }
            };
        });

    $(document).ready(
        function () {
            $('#voteCreator').on("click", function () {
                $('#creatorName').css("background", "green");
                $('#challengerName').css("background", "red");
            $.ajax({
                url: "/Debate/PlaceVote",
                type: "POST",
                data: {
                    debateId: '@Model.Debate.DebateId',
                    votePosition: 1
                }

            });
            $('#spectatorInformation').append('<br />' + "You have placed your vote on the
instigator!");
        });
        });

    $(document).ready(
    function () {
        $('#voteChallenger').on("click", function () {
            $('#challengerName').css("background", "green");
            $('#creatorName').css("background", "red");
            $.ajax({
```

```
                url: "/Debate/PlaceVote",
                type: "POST",
                data: {
                    debateId: '@Model.Debate.DebateId',
                    votePosition: 2
                }

            });
            $('#spectatorInformation').append('<br />' + "You have placed your vote on the
challenger!");
        });
    });

</script>



@{
    ViewBag.Title = "Index";
}


<div class="Headline">
    <p>FAQ</p>
</div>
<div>
    <table class="tableFAQ" >
        <ul>
            <tr>
                <td class="tableDataFAQ">
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate sad a sd",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
                </td>
            <td class="tableDataFAQ">
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            </td>
            <tr />


        </ul>
    </table>
</div>

<div class="Headline">
    <p>SUPPORT</p>
</div>
<div>
    <table class="tableFAQ">
        <ul>
            <tr>
                <td class="tableDataFAQ">
```

```
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            </td>
            <td class="tableDataFAQ">
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            <li class="li-header-size">@Html.ActionLink("Whats is Rate my Debate",
"WhatIsRateMyDebate", "FAQ") </li>
            </td>
            <tr />


        </ul>
    </table>
</div>


@{
    ViewBag.Title = "WhatIsRateMyDebate";
}

<div class="Headline">
    <p>What is rate my debate?</p>
</div>
<div class="tableFAQ" >
    <div class="textColor" style="text-align: center;">
        Rate my debate is a asæjdlksajdas
sdfffffffffffffffffffffffffffffffffffffffffffffffffffff
        dfsafsa
        fsafsa
        fasfasRate my debate is a asæjdlksajdas
sdfffffffffffffffffffffffffffffffffffffffffffffffffffff
        dfsafsa
        fsafsa
        fasfasRate my debate is a asæjdlksajdas
sdfffffffffffffffffffffffffffffffffffffffffffffffffffff
        dfsafsa
        fsafsa
        fasfas
    </div>
</div>



@{
    ViewBag.Title = "Index2";
}



<body align="center">

    <ul class="nav nav-tabs nav-justified" role="tablist">
```

```
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
        <li>@Html.ActionLink("Home", "Index", "Home")</li>
    </ul>

    <br />

    <div>
        <input value="User Name" />
    </div>
    <div>
        <input value="Password" style="margin-top:5px" />
    </div>

    <div>
        <button type="button" style="margin-top:5px; margin-left:0px">Create</button>
        <button type="button" style="margin-top: 5px; margin-right:37px">Log In</button>

    </div>
    <br />
    <div class="panel panel-default" style="margin-left:50px; margin-right:50px">
        <div class="panel-heading">
            <h3 class="panel-title">TEST</h3>
        </div>
        <div class="panel-body">
            Here goes the content!
        </div>

    </div>
</body>


@model RateMyDebate.Models.UserModel

@{
    ViewBag.Title = "Login";
}



@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="loginCreate">
        <div>
            @Html.LabelFor(model => model.userName)
            <div>
                @Html.TextBoxFor(model => model.userName)
                @Html.ValidationMessageFor(model => model.userName)

            </div>
        </div>
        <div>
            @Html.LabelFor(model => model.Password)
            <div>
                @Html.PasswordFor(model => model.Password)
                @Html.ValidationMessageFor(model => model.Password)
```

```
            </div>
        </div>

        @*<div>
         @Html.LabelFor(model => model.ConfirmPassword)
         <div>
             @Html.PasswordFor(model => model.ConfirmPassword)
             @Html.ValidationMessageFor(model => model.ConfirmPassword)
         </div>
    </div>
*@
         <div style="margin-top: 5px;">
             <input type="submit" value="Log in" class="btn btn-default" />
         </div>
    </div>



}


@{
    ViewBag.Title = "Index";
}
<div class="containerFrontDiv">
    <div class="startDiv" id="frontDiv">Start debating!</div>
    <table class="subDivTable">
        <tr>
            <td id="leftSideDiv">
                <div id="subFrontDiv">
                    Statistics <div id="subFrontDescription">Curious about who the top
debaters or what the hot topics are?<br/> Click here to see!</div>
                </div>
            </td>
            <td id="rightSideDiv">
                <div id="subFrontDiv">
                    Rules<div id="subFrontDescription">New to RateMyDebate? Please make sure
that you're familiar with the ruleset before beginning.</div>
                </div>
            </td>
        </tr>
        <tr>
            <td id="leftSideDiv">
                <div id="subFrontDiv">
                    FAQ/Support <div id="subFrontDescription">Please refer to this page, if
you have any questions in regards to using the site.</div>
                </div>
            </td>
            <td id="rightSideDiv">
                <div id="subFrontDiv">
                    Archive<div id="subFrontDescription">A searchable, sortable archive of
all debates conducted on the site. <br/> </div>
                </div>
            </td>
        </tr>
    </table>

</div>

<div class="tabs">
```

```
    <ul class="tab-links">
        <li class="active"><a href="#tab1">Popular debates</a></li>
        <li><a href="#tab2">Top debaters</a></li>
        <li><a href="#tab3">Random debates</a></li>
    </ul>

    <div class="tab-content">
        <div id="tab1" class="tab active">
            <table>
                <tr style="border-bottom: rgb(0, 0, 0, 0.25); font-weight: bold;">
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50)">Author</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50);">Challenger</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 55%;
background: rgba(167, 166, 255, 0.50);">Title</td>
                    <td style="padding-left: 100px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50)">Spectators</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.25); ">John</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.25); ">Peter</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 55%;
background: rgba(167, 166, 255, 0.25); ">Animal rights</td>
                    <td style="padding-left: 100px; width: 15%; background: rgba(167, 166,
255, 0.25); padding-right: 40px; ">320</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">Joffrey</td>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">Ned</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">
                        Off with his head?
                    </td>
                    <td style="padding-left: 100px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35); ">4000</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Ghost</td>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Busters</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Who you gonna call?</td>
                    <td style="padding-left: 100px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25)">6</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35)">Obama</td>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35)">McCain</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">
                        Why are you so grumpy, McCain?
                    </td>
```

```
                    <td style="padding-left: 100px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35); ">USA</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25); ">God</td>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25); ">Satan</td>
                    <td style="padding-left: 40px; width: 55; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">You started it</td>
                    <td style="padding-left: 100px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25)">6.000.000.000</td>
                </tr>
            </table>
        </div>

        <div id="tab2" class="tab">
            <table>
                <tr style="border-bottom: rgb(0, 0, 0, 0.25); font-weight: bold;">
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50)">Author</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50);">Challenger</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 55%;
background: rgba(167, 166, 255, 0.50);">Title</td>
                    <td style="padding-left: 100px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50)">Spectators</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.25); ">Santa</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.25); ">Peter</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 55%;
background: rgba(167, 166, 255, 0.25); ">Animal rights</td>
                    <td style="padding-left: 100px; width: 15%; background: rgba(167, 166,
255, 0.25); padding-right: 40px; ">320</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">Joffrey</td>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">Ned</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">
                        Off with his head?
                    </td>
                    <td style="padding-left: 100px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35); ">4000</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Ghost</td>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Busters</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Who you gonna call?</td>
                    <td style="padding-left: 100px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25)">6</td>
                </tr>
```

```
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35)">Obama</td>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35)">McCain</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">
                        Why are you so grumpy, McCain?
                    </td>
                    <td style="padding-left: 100px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35); ">USA</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25); ">God</td>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25); ">Satan</td>
                    <td style="padding-left: 40px; width: 55; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">You started it</td>
                    <td style="padding-left: 100px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25)">6.000.000.000</td>
                </tr>
            </table>
        </div>

        <div id="tab3" class="tab">
            <table>
                <tr style="border-bottom: rgb(0, 0, 0, 0.25); font-weight: bold;">
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50)">Author</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50);">Challenger</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 55%;
background: rgba(167, 166, 255, 0.50);">Title</td>
                    <td style="padding-left: 100px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.50)">Spectators</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.25); ">Mick</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 15%;
background: rgba(167, 166, 255, 0.25); ">Peter</td>
                    <td style="padding-left: 40px; padding-right: 40px; width: 55%;
background: rgba(167, 166, 255, 0.25); ">Animal rights</td>
                    <td style="padding-left: 100px; width: 15%; background: rgba(167, 166,
255, 0.25); padding-right: 40px; ">320</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">Joffrey</td>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">Ned</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">
                        Off with his head?
                    </td>
                    <td style="padding-left: 100px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35); ">4000</td>
                </tr>
                <tr>
```

```
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Ghost</td>
                    <td style="padding-left: 40px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Busters</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">Who you gonna call?</td>
                    <td style="padding-left: 100px; width: 15%; padding-right: 40px;
background: rgba(167, 166, 255, 0.25)">6</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35)">Obama</td>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35)">McCain</td>
                    <td style="padding-left: 40px; width: 55%; padding-right: 40px;
background: rgba(167, 166, 255, 0.35)">
                        Why are you so grumpy, McCain?
                    </td>
                    <td style="padding-left: 100px; padding-right: 40px; background:
rgba(167, 166, 255, 0.35); ">USA</td>
                </tr>
                <tr>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25); ">God</td>
                    <td style="padding-left: 40px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25); ">Satan</td>
                    <td style="padding-left: 40px; width: 55; padding-right: 40px;
background: rgba(167, 166, 255, 0.25); ">You started it</td>
                    <td style="padding-left: 100px; padding-right: 40px; background:
rgba(167, 166, 255, 0.25)">6.000.000.000</td>
                </tr>
            </table>
        </div>
    </div>
</div>


<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script type="text/javascript">
    jQuery(document).ready(function () {
        jQuery('.tabs .tab-links a').on('click', function (e) {
            var currentAttrValue = jQuery(this).attr('href');


            jQuery('.tabs ' + currentAttrValue).show().siblings().hide();


            jQuery(this).parent('li').addClass('active').siblings().removeClass('active');

            e.preventDefault();
        });
    });

    $(document).ready(function() {
        $('.startDiv').on("click", function () {
            window.location = "debate/index";
        });


    });
```

```
</script>

@model RateMyDebate.Models.Inbox

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Inbox</h4>
        <hr />
        @Html.ValidationSummary(true)

        <div class="form-group">
            @Html.LabelFor(model => model.userId, "userId", new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("userId", String.Empty)
                @Html.ValidationMessageFor(model => model.userId)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@model RateMyDebate.Models.Inbox

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Inbox</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.userInformation.fName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.userInformation.fName)
```

```
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

```
@model RateMyDebate.Models.Inbox

@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>Inbox</h4>
            <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.userInformation.fName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.userInformation.fName)
        </dd>

    </dl>
</div>
<p>
    @Html.ActionLink("Edit", "Edit", new { id = Model.inboxId }) |
    @Html.ActionLink("Back to List", "Index")
</p>
```

```
@model RateMyDebate.Models.Inbox

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Inbox</h4>
        <hr />
```

```
        @Html.ValidationSummary(true)
        @Html.HiddenFor(model => model.inboxId)

        <div class="form-group">
            @Html.LabelFor(model => model.userId, "userId", new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("userId", String.Empty)
                @Html.ValidationMessageFor(model => model.userId)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@model IEnumerable<RateMyDebate.Models.Inbox>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.userInformation.fName)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.userInformation.fName)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.inboxId }) |
            @Html.ActionLink("Details", "Details", new { id=item.inboxId }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.inboxId })
        </td>
    </tr>
}

</table>
```

```
@model RateMyDebate.Models.Message

@{
    ViewBag.Title = "Create";
}



@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4 style="margin-left: 20px; margin-top: 20px;">Send Message</h4>
        <hr />
        @Html.ValidationSummary(true)
@*          <div class="form-group">
        @Html.LabelFor(model => model.userId, "userId", new { @class = "control-label col-
md-2" })
        <div class="col-md-10">
            @Html.DropDownList("userId", String.Empty)
            @Html.ValidationMessageFor(model => model.userId)
        </div>
    </div>


        <div class="form-group">
            @Html.LabelFor(model => model.inboxKey, "inboxKey", new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("inboxKey", String.Empty)
                @Html.ValidationMessageFor(model => model.inboxKey)
            </div>
        </div>
*@
        <div class="form-group">
            @Html.LabelFor(model => model.Receiver, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.Receiver)
                @Html.ValidationMessageFor(model => model.Receiver)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.subject, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.subject)
                @Html.ValidationMessageFor(model => model.subject)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.messageText, new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
```

```
                @Html.TextAreaFor(model => model.messageText, new { style = "width:500px;
height: 300px;" })
                @Html.ValidationMessageFor(model => model.messageText)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@model RateMyDebate.Models.Message

@{
    ViewBag.Title = "Delete";
}



<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Message</h4>
    <hr />
    <dl class="dl-horizontal">


        <dt>
            @Html.DisplayNameFor(model => model.userInformation.fName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.userInformation.fName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.subject)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.subject)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.messageText)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.messageText)
        </dd>

    </dl>
```

```
    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to inbox", "Index")
        </div>
    }
</div>


@model RateMyDebate.Models.Message

@{
    ViewBag.Title = "Details";
}



<div>
    <h4 style="text-align: center;">Message</h4>
          <hr />
    <dl class="loginCreate">


        <dt>
            @Html.DisplayNameFor(model => model.userInformation.fName):
        </dt>

        <dd>
            @Html.DisplayFor(model => model.userInformation.fName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.subject):
        </dt>

        <dd>
            @Html.DisplayFor(model => model.subject)
        </dd>
        <hr />
        <dt>
            @Html.DisplayNameFor(model => model.messageText):
        </dt>

        <dd>
            @Html.DisplayFor(model => model.messageText)
        </dd>

    </dl>
</div>
<p>
    @* @Html.ActionLink("Edit", "Edit", new { id = Model.messageId }) |*@
    @Html.ActionLink("Back to inbox", "Index")
</p>


@model RateMyDebate.Models.Message

@{
```

```
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Message</h4>
        <hr />
        @Html.ValidationSummary(true)
        @Html.HiddenFor(model => model.messageId)

        <div class="form-group">
            @Html.LabelFor(model => model.userId, "userId", new { @class = "control-label
col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("userId", String.Empty)
                @Html.ValidationMessageFor(model => model.userId)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.inboxKey, "inboxKey", new { @class = "control-
label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownList("inboxKey", String.Empty)
                @Html.ValidationMessageFor(model => model.inboxKey)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.subject, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.subject)
                @Html.ValidationMessageFor(model => model.subject)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.messageText, new { @class = "control-label col-md-
2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.messageText)
                @Html.ValidationMessageFor(model => model.messageText)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
```

```
    @Html.ActionLink("Back to List", "Index")
</div>


@model IEnumerable<RateMyDebate.Models.Message>

@{
    ViewBag.Title = "Index";
}

<h2 style="margin-left: 20px;">Messages</h2>


<div class="loginCreate" style="width: 96%;">
<table class="table">
    <tr>
        <button type="button" style="margin-bottom:20px; padding:3px;"
onclick="redirect()">Send Message</button>
        <th>
            @Html.DisplayNameFor(model => model.Sender)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.subject)
        </th>

        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>

        <td>
            @Html.DisplayFor(modelItem => item.Sender)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.subject)
        </td>

        <td>

            @Html.ActionLink("View Message", "Details", new { id=item.messageId }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.messageId })
        </td>
    </tr>
}

</table>
</div>

<script type="text/javascript">
    function redirect() {
        location.href = '@Url.Content("~/Message/Create/")';
    }
</script>


@{
    ViewBag.Title = "NotLoggedIn";
}
```

```
<h2>Please log in before viewing messages</h2>



@{
    ViewBag.Title = "Index";
}

<div class="Headline">
    <p>Rules!</p>
</div>

<div class="tableFAQ">
    <div class="textColor" style=" text-align: center;">
        <p class="Headline">Follow the rules on the site or GTFO.</p>

        <p class="Headline"> Rule 1.</p>
        sdfsdfsdfsddddddddddddd sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsd

        <p class="Headline"> Rule 2.</p>
        sdfsdfsdfsddddddddddddd sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsd

        <p class="Headline"> Rule 3.</p>
        sdfsdfsdfsddddddddddddd sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsd
        <p class="Headline"> Rule 4.</p>
        sdfsdfsdfsddddddddddddd sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddd
        sdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsddddddddddddsdfsdfsdfsd

    </div>

</div>


@using System.Web.UI.WebControls
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title</title>
    @Styles.Render("~/Content/css")
    @Styles.Render("~/Content/RMD.css")
    @Scripts.Render("~/bundles/modernizr")
```

```
</head>
<body>
    <div class="mainDiv">
        <div class="bannerDiv">

           <img src="~/imgs/logo.png" />
            @*<img src="~/imgs/logotitle.png" />*@

               <div class="loginDiv">

                @if (Request.IsAuthenticated)
                {
                    <strong>@Html.Encode(User.Identity.Name)</strong>
                    @Html.ActionLink("Log out", "LogOut", "Home")
                    @Html.ActionLink("Profile", "Profile", "UserInformation")
                    @Html.ActionLink("Messages", "Index", "Message")

                   // UserModel userSession = Session["userSession"] as UserModel;
                }
                else
                {

                    @Html.ActionLink("Login","Login","Home") <br />
                    @Html.ActionLink("Register", "Create", "User")
                }
            @*
                <form>
                    Username: <input type="text" name="username" /><br/>
                    Password:  <input type="text" name="password" />
                    <br /><input class="loginButton" type="submit" value="Login!" />
                    <div class="loginText"><a> @Html.ActionLink("Register", "Create",
"User")</a> | <a>Forgot password?</a></div>
                </form>
            *@
    @* @Html.Partial("_LoginPartial"); *@
            </div>
            <ul class="nav nav-tabs nav-justified" style="margin-top: 2px;" role="tablist">
                <li> @Html.ActionLink("Home", "RasmusIndex", "Home")</li>
                <li> @Html.ActionLink("Statistics", "Index", "Statistics")</li>
                <li> @Html.ActionLink("Rules", "Index", "Rules")</li>
                <li> @Html.ActionLink("FAQ/Support", "Index", "FAQ")</li>
            </ul>

        </div>
        <div style="background: white ; padding-top: 40px; padding-bottom: 0; margin: 0">

            @RenderBody()

            @*Temp løsning*@
            <br /><br /><br />
            <div class="bottomDiv">
                This is a footer for small details
            </div>
        </div>
    </div>
    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

```
@using (Html.BeginForm())
{
    <p>
        Category: @Html.DropDownList("category", "All")
        Creator: @Html.TextBox("creator")
        Challenger: @Html.TextBox("challenger")
        <input type="submit" value="Filter" />
    </p>
}
```

```
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "Error";
}

<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error occurred while processing your request.</h2>
```

```
@model RateMyDebate.Models.UserModel

@{
    ViewBag.Title = "Create";
}
```

```
<link href="~/Content/Site.css" rel="stylesheet" />
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

<script src="@Url.Content("~/Scripts/jquery.validate.min.js")"
type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>

@using (Html.BeginForm())
{
    @Html.ValidationSummary(true);
    @Html.AntiForgeryToken()
    <div class="loginCreate">
        <h4>Login Details</h4>
        <div>
            @Html.LabelFor(model => model.userName)
            <div>
                @Html.TextBoxFor(model => model.userName)
                @Html.ValidationMessageFor(model => model.userName)
            </div>
        </div>
        <div>
            @Html.LabelFor(model => model.Password)
            <div>
                @Html.PasswordFor(model => model.Password)
```

```
                @Html.ValidationMessageFor(model => model.Password)
            </div>
        </div>
        <div>
            @Html.LabelFor(model => model.ConfirmPassword)
            <div>
                @Html.PasswordFor(model => model.ConfirmPassword)
                @Html.ValidationMessageFor(model => model.ConfirmPassword)
            </div>
        </div>

        <div style ="margin-top: 5px;">
            <input type="submit" value="Save" class="btn btn-default" />
        </div>
    </div>
}

@model RateMyDebate.Models.UserModel

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>UserModel</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.userName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.userName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Password)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Password)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>


@model RateMyDebate.Models.UserModel
```

```
@{
    ViewBag.Title = "Details";
}

<h2>Details</h2>

<div>
    <h4>UserModel</h4>
            <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.userName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.userName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Password)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Password)
        </dd>

    </dl>
</div>
<p>

    @Html.ActionLink("Back to List", "Index")
</p>


@model RateMyDebate.Models.UserModel

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>UserModel</h4>
        <hr />
        @Html.ValidationSummary(true)
        @Html.HiddenFor(model => model.accountId)

        <div class="form-group">
            @Html.LabelFor(model => model.userName, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.userName)
                @Html.ValidationMessageFor(model => model.userName)
            </div>
```

```
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Password, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.Password)
                @Html.ValidationMessageFor(model => model.Password)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@model IEnumerable<RateMyDebate.Models.UserModel>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.userName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Password)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.userName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Password)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.accountId }) |
            @Html.ActionLink("Details", "Details", new { id=item.accountId }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.accountId })
        </td>
    </tr>
```

```
}

</table>


@model RateMyDebate.Models.UserInformation

@{
    ViewBag.Title = "Create";
}

<link href="~/Content/Site.css" rel="stylesheet" />
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

<script src="@Url.Content("~/Scripts/jquery.validate.min.js")"
type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>

@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <br />
        <h4>User Details</h4>
        <hr />
        @Html.ValidationSummary(true)

        <div class="form-group">
            @Html.LabelFor(model => model.fName, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.fName)
                @Html.ValidationMessageFor(model => model.fName)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.lName, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.lName)
                @Html.ValidationMessageFor(model => model.lName)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.nickName, new { @class = "control-label col-md-2"
})
            <div class="col-md-10">
                @Html.EditorFor(model => model.nickName)
                @Html.ValidationMessageFor(model => model.nickName)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.age, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.age)
```

```
                @Html.ValidationMessageFor(model => model.age)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.autobiography, new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.TextAreaFor(model => model.autobiography, new { style = "width:500px;
height: 300px;" })
                @Html.ValidationMessageFor(model => model.autobiography)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email)
                @Html.ValidationMessageFor(model => model.Email)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@model RateMyDebate.Models.UserInformation

@{
    ViewBag.Title = "Delete";
}

<h2>Delete</h2>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>UserInformation</h4>
    <hr />
    <dl class="dl-horizontal">
        <dt>
            @Html.DisplayNameFor(model => model.fName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.fName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.lName)
        </dt>
```

```
        <dd>
            @Html.DisplayFor(model => model.lName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.nickName)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.nickName)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.age)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.age)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.autobiography)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.autobiography)
        </dd>

        <dt>
            @Html.DisplayNameFor(model => model.Email)
        </dt>

        <dd>
            @Html.DisplayFor(model => model.Email)
        </dd>

    </dl>

    @using (Html.BeginForm()) {
        @Html.AntiForgeryToken()

        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-default" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>


@model RateMyDebate.Models.UserInformation

@{
    ViewBag.Title = "Details";
}



<br />
<div class="tableFAQ">
    <div class="Headline">
```

```
    <h3>Profile: @Model.nickName User foreignKey @Model.userId</h3>
</div>
    <p style="text-align: right; padding-right: 10px; margin-top: -30px;">
        @Html.ActionLink("Edit", "Edit", new { id = Model.userInformationId })
    </p>

<hr />

<div class="detailFormat">

    <table style="width:200px">
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.fName)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.fName)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.lName)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.lName)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.nickName)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.nickName)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.age)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.age)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.Email)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.Email)
            </td>
```

```
            </tr>

        </table>

    </div>
    <hr />
    <div class="autobiographyFormat">
        <div>
            <b>
                @Html.DisplayNameFor(model => model.autobiography)
            </b>
        </div>
        @Html.DisplayFor(model => model.autobiography)
    </div>

</div>



@model RateMyDebate.Models.UserInformation

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>
<link href="~/Content/Site.css" rel="stylesheet" />
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery-1.10.2.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>

<script src="@Url.Content("~/Scripts/jquery.validate.min.js")"
type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    @Html.HiddenFor(model => model.userId)
    @Html.HiddenFor(model => model.nickName)

    <div class="form-horizontal">
        <h4>UserInformation</h4>
        <hr />
        @Html.ValidationSummary(true)
        @Html.HiddenFor(model => model.userInformationId)

        <div class="form-group">
            @Html.LabelFor(model => model.fName, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.fName)
                @Html.ValidationMessageFor(model => model.fName)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.lName, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.lName)
                @Html.ValidationMessageFor(model => model.lName)
```

```
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.age, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.age)
                @Html.ValidationMessageFor(model => model.age)
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.autobiography, new { @class = "control-label col-
md-2" })
            <div class="col-md-10">
                @Html.TextAreaFor(model => model.autobiography, new { style = "width:500px;
height: 300px;" })
                @Html.ValidationMessageFor(model => model.autobiography)
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.Email, new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email)
                @Html.ValidationMessageFor(model => model.Email)
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default" />
            </div>
        </div>
    </div>

}

<div>
   @Html.ActionLink("Change Password", "Edit2", new { id = Model.userId })
</div>


@model RateMyDebate.Models.UserModel

@{
    ViewBag.Title = "Edit";
}



@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="loginCreate" >
        <div class="form-horizontal">
            <h4>Change Password!</h4>
            <hr />
            @Html.ValidationSummary(true)
            @Html.HiddenFor(model => model.accountId)
            @Html.HiddenFor(model => model.userName)
```

```
            <div>
                @Html.LabelFor(model => model.Password)
                <div >
                    @Html.PasswordFor(model => model.Password)
                    @Html.ValidationMessageFor(model => model.Password)
                </div>
            </div>
            <div >
                @Html.LabelFor(model => model.ConfirmPassword)
                <div >
                    @Html.PasswordFor(model => model.ConfirmPassword)
                    @Html.ValidationMessageFor(model => model.ConfirmPassword)
                </div>
            </div>

            <div >
                <div style="margin-top: 5px;">
                    <input type="submit" value="Save" class="btn btn-default" />
                </div>
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>


@model IEnumerable<RateMyDebate.Models.UserInformation>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.fName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.lName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.nickName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.age)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.autobiography)
        </th>
        <th>
```

```
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.userId)
        </th>
        <th></th>
    </tr>

@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.fName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.lName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.nickName)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.age)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.autobiography)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Email)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.userId)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.userInformationId }) |
            @Html.ActionLink("Details", "Details", new { id=item.userInformationId }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.userInformationId })
        </td>
    </tr>
}

</table>


@model RateMyDebate.Models.UserInformation

@{
    ViewBag.Title = "Profile";
}



<br />
<div class="tableFAQ">
    <div class="Headline">
        <h3>Profile: @Model.nickName User foreignKey @Model.userId</h3>
    </div>
    <p style="text-align: right; padding-right: 10px; margin-top: -30px;">
        @Html.ActionLink("Edit", "Edit", new { id = Model.userInformationId })<br />
        @Html.ActionLink("Search for users", "Profiles")
    </p>
```

```
<hr />

<div class="detailFormat">

    <table style="width:250px">
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.fName)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.fName)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.lName)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.lName)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.nickName)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.nickName)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.age)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.age)
            </td>
        </tr>
        <tr>
            <td>
                <b>
                    @Html.DisplayNameFor(model => model.Email)
                </b>
            </td>
            <td>
                @Html.DisplayFor(model => model.Email)
            </td>
        </tr>

    </table>

</div>
<hr />
```

```
    <div class="autobiographyFormat">
        <div>
            <b>
                @Html.DisplayNameFor(model => model.autobiography)
            </b>
        </div>
        @Html.DisplayFor(model => model.autobiography)
    </div>

</div>



@model IEnumerable<RateMyDebate.Models.UserInformation>

@{
    ViewBag.Title = "Profiles";
}


@*
    <br />
    <div class="tableFAQ">
        <div class="Headline">
            <h3>Profile: @Model.nickName User foreignKey @Model.userId</h3>
        </div>


        <hr />

        <div class="detailFormat">

            <table style="width:250px">
                <tr>
                    <td>
                        <b>
                            @Html.DisplayNameFor(model => model.fName)
                        </b>
                    </td>
                    <td>
                        @Html.DisplayFor(model => model.fName)
                    </td>
                </tr>
                <tr>
                    <td>
                        <b>
                            @Html.DisplayNameFor(model => model.lName)
                        </b>
                    </td>
                    <td>
                        @Html.DisplayFor(model => model.lName)
                    </td>
                </tr>
                <tr>
                    <td>
                        <b>
                            @Html.DisplayNameFor(model => model.nickName)
                        </b>
                    </td>
                    <td>
```

```
                    @Html.DisplayFor(model => model.nickName)
                </td>
            </tr>
            <tr>
                <td>
                    <b>
                        @Html.DisplayNameFor(model => model.age)
                    </b>
                </td>
                <td>
                    @Html.DisplayFor(model => model.age)
                </td>
            </tr>
            <tr>
                <td>
                    <b>
                        @Html.DisplayNameFor(model => model.Email)
                    </b>
                </td>
                <td>
                    @Html.DisplayFor(model => model.Email)
                </td>
            </tr>

        </table>

    </div>
    <hr />
    <div class="autobiographyFormat">
        <div>
            <b>
                @Html.DisplayNameFor(model => model.autobiography)
            </b>
        </div>
        @Html.DisplayFor(model => model.autobiography)
    </div>

</div>

*@
@using (Html.BeginForm())
{
    <div class="loginCreate">
        <p>
            Nickname: @Html.TextBox("SearchName") <br />

            <input type="submit" value="Search" />
        </p>
        <div style="color: red;">
            @ViewBag.NotFound
        </div>
    </div>
}
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.fName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.lName)
```

```
        </th>
        <th>
            @Html.DisplayNameFor(model => model.nickName)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.age)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.autobiography)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.userId)
        </th>
        <th></th>
    </tr>

    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.fName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.lName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.nickName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.age)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.autobiography)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Email)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.userId)
            </td>
            <td>

                @Html.ActionLink("Profile", "Details", new { id = item.userInformationId })

            </td>
        </tr>

    }
</table>


@{
    ViewBag.Title = "SearchView";
}

<h2>SearchView</h2>
```

```
@using (Html.BeginForm())
{
    <div class="loginCreate">
        <p>
            Nickname: @Html.TextBox("SearchName") <br />

            <input type="submit" value="Search" />
        </p>
        <div style="color: red;">
            @ViewBag.NotFound
        </div>
    </div>
}


@{
    ViewBag.Title = "UserEditError";
}

<h2>You can not edit that user!</h2>



@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}


@model RateMyDebate.Models.Debate

@{
    Layout = null;
}

<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>StartDebate</title>
</head>
<body>
    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()

        <div class="form-horizontal">
            <h4>Debate</h4>
            <hr />
            @Html.ValidationSummary(true)

            <div class="form-group">
                @Html.LabelFor(model => model.CreatorIdId, "CreatorIdId", new { @class =
"control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownList("CreatorIdId", String.Empty)
                    @Html.ValidationMessageFor(model => model.CreatorIdId)
                </div>
            </div>
```

```
            <div class="form-group">
                @Html.LabelFor(model => model.ChallengerIdId, "ChallengerIdId", new { @class
= "control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownList("ChallengerIdId", String.Empty)
                    @Html.ValidationMessageFor(model => model.ChallengerIdId)
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.CategoryIdId, "CategoryIdId", new { @class =
"control-label col-md-2" })
                <div class="col-md-10">
                    @Html.DropDownList("CategoryIdId", String.Empty)
                    @Html.ValidationMessageFor(model => model.CategoryIdId)
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Subject, new { @class = "control-label col-md-
2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Subject)
                    @Html.ValidationMessageFor(model => model.Subject)
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Description, new { @class = "control-label
col-md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Description)
                    @Html.ValidationMessageFor(model => model.Description)
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.ChatText, new { @class = "control-label col-
md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.ChatText)
                    @Html.ValidationMessageFor(model => model.ChatText)
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.Live, new { @class = "control-label col-md-2"
})
                <div class="col-md-10">
                    @Html.EditorFor(model => model.Live)
                    @Html.ValidationMessageFor(model => model.Live)
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.DateTime, new { @class = "control-label col-
md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.DateTime)
                    @Html.ValidationMessageFor(model => model.DateTime)
```

```
                </div>
            </div>

            <div class="form-group">
                @Html.LabelFor(model => model.TimeLimit, new { @class = "control-label col-
md-2" })
                <div class="col-md-10">
                    @Html.EditorFor(model => model.TimeLimit)
                    @Html.ValidationMessageFor(model => model.TimeLimit)
                </div>
            </div>

            <div class="form-group">
                <div class="col-md-offset-2 col-md-10">
                    <input type="submit" value="Create" class="btn btn-default" />
                </div>
            </div>
        </div>
    }

    <div>
        @Html.ActionLink("Back to List", "Index")
    </div>
</body>
</html>


<?xml version="1.0"?>

<configuration>
  <configSections>
    <sectionGroup name="system.web.webPages.razor"
type="System.Web.WebPages.Razor.Configuration.RazorWebSectionGroup,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
      <section name="host" type="System.Web.WebPages.Razor.Configuration.HostSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
      <section name="pages" type="System.Web.WebPages.Razor.Configuration.RazorPagesSection,
System.Web.WebPages.Razor, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false" />
    </sectionGroup>
  </configSections>

  <system.web.webPages.razor>
    <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc,
Version=5.0.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35" />
    <pages pageBaseType="System.Web.Mvc.WebViewPage">
      <namespaces>
        <add namespace="System.Web.Mvc" />
        <add namespace="System.Web.Mvc.Ajax" />
        <add namespace="System.Web.Mvc.Html" />
        <add namespace="System.Web.Optimization"/>
        <add namespace="System.Web.Routing" />
        <add namespace="RateMyDebate" />
      </namespaces>
    </pages>
  </system.web.webPages.razor>

  <appSettings>
```

```
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>

  <system.webServer>
    <handlers>
      <remove name="BlockViewHandler"/>
      <add name="BlockViewHandler" path="*" verb="*" preCondition="integratedMode"
type="System.Web.HttpNotFoundHandler" />
    </handlers>
  </system.webServer>
</configuration>
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
using Ninject;
using RateMyDebate.Models;

namespace RateMyDebate
{
    public class ControllerFactory : DefaultControllerFactory
    {

        private IKernel Kernel;
        public ControllerFactory()
        {
            Kernel = new StandardKernel();
            AddBindings();
        }
        protected override IController GetControllerInstance(RequestContext requestContext,
Type controllerType)
        {
            return controllerType == null ? null : (IController)Kernel.Get(controllerType);
        }
        private void AddBindings()
        {
            Kernel.Bind<IDebateRepository>().To<DebateRepository>();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;

namespace RateMyDebate
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
```

```
        {
            AreaRegistration.RegisterAllAreas();
            FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            BundleConfig.RegisterBundles(BundleTable.Bundles);

            ControllerBuilder.Current.SetControllerFactory(new ControllerFactory());
        }
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Antlr" version="3.4.1.9004" targetFramework="net45" />
  <package id="bootstrap" version="3.0.0" targetFramework="net45" />
  <package id="EntityFramework" version="6.1.1" targetFramework="net45" />
  <package id="jQuery" version="1.10.2" targetFramework="net45" />
  <package id="jQuery.Validation" version="1.11.1" targetFramework="net45" />
  <package id="Microsoft.AspNet.Identity.Core" version="1.0.0" targetFramework="net45" />
  <package id="Microsoft.AspNet.Identity.EntityFramework" version="1.0.0"
targetFramework="net45" />
  <package id="Microsoft.AspNet.Identity.Owin" version="1.0.0" targetFramework="net45" />
  <package id="Microsoft.AspNet.Mvc" version="5.2.2" targetFramework="net45" />
  <package id="Microsoft.AspNet.Razor" version="3.2.2" targetFramework="net45" />
  <package id="Microsoft.AspNet.SignalR" version="2.1.1" targetFramework="net45" />
  <package id="Microsoft.AspNet.SignalR.Core" version="2.1.1" targetFramework="net45" />
  <package id="Microsoft.AspNet.SignalR.JS" version="2.1.1" targetFramework="net45" />
  <package id="Microsoft.AspNet.SignalR.SystemWeb" version="2.1.1" targetFramework="net45"
/>
  <package id="Microsoft.AspNet.Web.Optimization" version="1.1.1" targetFramework="net45" />
  <package id="Microsoft.AspNet.WebPages" version="3.2.2" targetFramework="net45" />
  <package id="Microsoft.jQuery.Unobtrusive.Validation" version="3.0.0"
targetFramework="net45" />
  <package id="Microsoft.Owin" version="2.0.2" targetFramework="net45" />
  <package id="Microsoft.Owin.Host.SystemWeb" version="2.0.2" targetFramework="net45" />
  <package id="Microsoft.Owin.Security" version="2.0.2" targetFramework="net45" />
  <package id="Microsoft.Owin.Security.Cookies" version="2.0.0" targetFramework="net45" />
  <package id="Microsoft.Owin.Security.Facebook" version="2.0.0" targetFramework="net45" />
  <package id="Microsoft.Owin.Security.Google" version="2.0.0" targetFramework="net45" />
  <package id="Microsoft.Owin.Security.MicrosoftAccount" version="2.0.0"
targetFramework="net45" />
  <package id="Microsoft.Owin.Security.OAuth" version="2.0.0" targetFramework="net45" />
  <package id="Microsoft.Owin.Security.Twitter" version="2.0.0" targetFramework="net45" />
  <package id="Microsoft.Web.Infrastructure" version="1.0.0.0" targetFramework="net45" />
  <package id="Modernizr" version="2.6.2" targetFramework="net45" />
  <package id="Moq" version="4.2.1409.1722" targetFramework="net45" />
  <package id="Newtonsoft.Json" version="5.0.6" targetFramework="net45" />
  <package id="Owin" version="1.0" targetFramework="net45" />
  <package id="Respond" version="1.2.0" targetFramework="net45" />
  <package id="SimpleCrypto" version="0.3.30.26" targetFramework="net45" />
  <package id="WebGrease" version="1.5.2" targetFramework="net45" />
</packages>
```

```
using Microsoft.Owin;
using Owin;

[assembly: OwinStartupAttribute(typeof(RateMyDebate.Startup))]
namespace RateMyDebate
{
```

```
    public partial class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            app.MapSignalR();

            ConfigureAuth(app);
        }
    }
}


<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=301880
  -->
<configuration>
  <configSections>

    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"
/>
  <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 --></configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\v11.0;AttachDbFilename=|DataDirectory|\aspnet-RateMyDebate-
20140815031513.mdf;Initial Catalog=aspnet-RateMyDebate-20140815031513;Integrated
Security=True" providerName="System.Data.SqlClient" />
  </connectionStrings>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
  <system.web>

      <authentication mode="Forms">
        <forms loginUrl="~/Home/Login" timeout="2880" />
      </authentication>

    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
  </system.web>
  <system.webServer>
    <modules>
      <remove name="FormsAuthenticationModule" />
    </modules>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">



      <dependentAssembly>
        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="0.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
```

```
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-2.0.2.0" newVersion="2.0.2.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-2.0.2.0" newVersion="2.0.2.0" />
      </dependentAssembly>
    <dependentAssembly><assemblyIdentity name="System.Web.Helpers"
publicKeyToken="31bf3856ad364e35" /><bindingRedirect oldVersion="1.0.0.0-3.0.0.0"
newVersion="3.0.0.0" /></dependentAssembly><dependentAssembly><assemblyIdentity
name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" /><bindingRedirect
oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0"
/></dependentAssembly><dependentAssembly><assemblyIdentity name="System.Web.Mvc"
publicKeyToken="31bf3856ad364e35" /><bindingRedirect oldVersion="1.0.0.0-5.2.2.0"
newVersion="5.2.2.0" /></dependentAssembly></assemblyBinding>
  </runtime>
  <entityFramework>
    <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
      <parameters>
        <parameter value="v11.0" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
</configuration>
```

```
[assembly:
WebActivatorEx.PreApplicationStartMethod(typeof(RateMyDebateTests.App_Start.NinjectWebCommon
), "Start")]
[assembly:
WebActivatorEx.ApplicationShutdownMethodAttribute(typeof(RateMyDebateTests.App_Start.Ninject
WebCommon), "Stop")]

namespace RateMyDebateTests.App_Start
{
    using System;
    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;
    using Ninject.Web.Common;

    public static class NinjectWebCommon
    {
        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>
        /// Starts the application
        /// </summary>
        public static void Start()
        {
```

```
            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));
            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));
            bootstrapper.Initialize(CreateKernel);
        }

        /// <summary>
        /// Stops the application.
        /// </summary>
        public static void Stop()
        {
            bootstrapper.ShutDown();
        }

        /// <summary>
        /// Creates the kernel that will manage your application.
        /// </summary>
        /// <returns>The created kernel.</returns>
        private static IKernel CreateKernel()
        {
            var kernel = new StandardKernel();
            try
            {
                kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new
Bootstrapper().Kernel);
                kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

                RegisterServices(kernel);
                return kernel;
            }
            catch
            {
                kernel.Dispose();
                throw;
            }
        }

        /// <summary>
        /// Load your modules or register your services here!
        /// </summary>
        /// <param name="kernel">The kernel.</param>
        private static void RegisterServices(IKernel kernel)
        {
        }
    }
}


<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit
http://go.microsoft.com/fwlink/?LinkID=237468 -->
    <section name="entityFramework"
type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework,
Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"
/>
  </configSections>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
```

```
        <assemblyIdentity name="Microsoft.Owin" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-2.0.2.0" newVersion="2.0.2.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="Microsoft.Owin.Security" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-2.0.2.0" newVersion="2.0.2.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="WebGrease" publicKeyToken="31bf3856ad364e35"
culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-1.5.2.14234" newVersion="1.5.2.14234" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
  <entityFramework>
    <defaultConnectionFactory
type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">
      <parameters>
        <parameter value="v11.0" />
      </parameters>
    </defaultConnectionFactory>
    <providers>
      <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    </providers>
  </entityFramework>
</configuration>
```

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="EntityFramework" version="6.1.1" targetFramework="net45" />
  <package id="Microsoft.AspNet.Mvc" version="5.2.2" targetFramework="net45" />
  <package id="Microsoft.AspNet.Razor" version="3.2.2" targetFramework="net45" />
  <package id="Microsoft.AspNet.WebPages" version="3.2.2" targetFramework="net45" />
  <package id="Microsoft.Web.Infrastructure" version="1.0.0.0" targetFramework="net45" />
  <package id="Moq" version="4.2.1409.1722" targetFramework="net45" />
  <package id="Ninject" version="3.2.2.0" targetFramework="net45" />
  <package id="Ninject.MVC5" version="3.2.1.0" targetFramework="net45" />
  <package id="Ninject.Web.Common" version="3.2.3.0" targetFramework="net45" />
  <package id="Ninject.Web.Common.WebHost" version="3.2.3.0" targetFramework="net45" />
  <package id="WebActivatorEx" version="2.0.6" targetFramework="net45" />
</packages>
```

```
using System;
using System.Web.Mvc;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using RateMyDebate.Controllers;
using RateMyDebate.Models;
using RateMyDebate.ViewModels;
using System.Web.WebPages;

namespace RateMyDebateTests
{
    [TestClass]
    public class UnitTest1
    {
        private readonly IDebateRepository debateRepository;
```

```
        [TestMethod]
        public void DebateTest()
        {
            DebateController controller = new DebateController(debateRepository);

            DebateDisplayViewModel result = controller.FindDebateDisplayViewModel(1) as
DebateDisplayViewModel;

            Assert.AreEqual(1, result.Debate.CreatorIdId);
        }

        [TestMethod]
        public void DebateTest2()
        {
            DebateController controller = new DebateController(debateRepository);

            var result = controller.LiveChat(1) as ViewResult;

            Assert.AreEqual("Cake", result.ViewBag.Title);
        }

        [TestMethod]
        public void DebateDisplayTest()
        {
            DebateController controller = new DebateController(debateRepository);

            ViewResult result = controller.Display(1) as ViewResult;

            Assert.IsNotNull(result);
        }

        [TestMethod]
        public void DebateDisplayTest2()
        {
            DebateController controller = new DebateController(debateRepository);

            var result = controller.Display(1) as ViewResult;

            var model = result.ViewData.Model as DebateDisplayViewModel;

            Assert.AreEqual(1, model.Debate.DebateId);
        }

        // Checking to make sure that debate list is not empty, when no filter parameters
are parsed and database is not empty.
        [TestMethod]
        public void DebateIndexTest()
        {
            DebateController controller = new DebateController(debateRepository);

            var result = controller.Index(null, null, null) as ViewResult;

            var model = result.ViewData.Model as DebateUser;

            var list = model.Debate;

            Assert.AreNotEqual(0, list.Count); ;
        }
```

```
// Testing to make sure the debate list is NOT empty,
// when parsing through a search keyword that we know to exist in the database.
[TestMethod]
public void DebateIndexTest2()
{
    DebateController controller = new DebateController(debateRepository);

    var result = controller.Index("Atheism", null, null) as ViewResult;

    var model = result.ViewData.Model as DebateUser;

    var list = model.Debate;

    Assert.AreNotEqual(0, list.Count);
}


// Testing to make sure the debate list contains 0 elements,
// when parsing through a search keyword that we know not to exist in the database.
[TestMethod]
public void DebateIndexTest3()
{
    DebateController controller = new DebateController(debateRepository);

    var result = controller.Index(null, "asdasdasdas", null) as ViewResult;

    var model = result.ViewData.Model as DebateUser;

    var list = model.Debate;

    Assert.AreEqual(0, list.Count);
}

[TestMethod]
public void DebateProcessingTest()
{
    DebateController controller = new DebateController(debateRepository);
    Debate debate = controller.FindDebate(1);
    int debateId = debate.DebateId;

    controller.ProcessDebateResult(debateId);

    Result result = controller.FindResult(debateId);

    int? winnerId = result.WinnerId;

    Assert.AreEqual(1, winnerId);
}

[TestMethod]
public void DebateProcessingTest2()
{

    DebateController controller = new DebateController(debateRepository);
    Debate debate = controller.FindDebate(2);
    int debateId = debate.DebateId;

    controller.ProcessDebateResult(debateId);

    Result result = controller.FindResult(debateId);
```

```
            int? winnerId = result.WinnerId;

            Assert.AreEqual(2, winnerId);
        }

        [TestMethod]
        public void DebateInactiveTest()
        {
            DebateController controller = new DebateController(debateRepository);

            int debateId = 1;

            controller.SetDebateInactive(1);

            Debate debate = controller.FindDebate(1);

            Assert.IsFalse(debate.Live);
        }
    }
}


using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using RateMyDebate.Controllers;
using RateMyDebate.Models;

namespace RateMyDebateTests
{
    [TestClass]
    public class UnitTest2
    {
        private readonly IDebateRepository debateRepository;
        [TestMethod]
        public void VoteTest1()
        {
            DebateController controller = new DebateController(debateRepository);

            Vote vote = controller.FindVote(9, 1);

            int testPos = vote.VotePos;

            Assert.AreEqual(1, testPos);
        }

        [TestMethod]
        public void VoteTest2()
        {
            DebateController controller = new DebateController(debateRepository);

            Vote vote = controller.FindVote(9, 2);

            int testPos = vote.VotePos;

            Assert.AreEqual(2, testPos);
        }

        [TestMethod]
```

```
        public void VoteTest3()
        {
            DebateController controller = new DebateController(debateRepository);

            Vote vote = controller.FindVote(8, 2);

            int testPos = vote.VotePos;

            Assert.AreEqual(1, testPos);
        }
    }
}
using System;
using System.Web;
using System.Web.ModelBinding;
using System.Web.Mvc;
using System.Web.SessionState;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Moq;
using RateMyDebate.Controllers;
using RateMyDebate.Models;
using System.Web.Security;

namespace RateMyDebateTests
{
    [TestClass]
    public class UnitTest3
    {
        [TestMethod]
        public void TestMethod1()
        {
            var mockRepo = new Mock<IDebateRepository>();

            DebateController controller = new DebateController(mockRepo.Object);

            var debate = new Debate
            {
                DebateId = 0, CategoryIdId = 4, CreatorIdId = 3, Live = true,
                Subject = "Emne", Description = "Beskrivelse",
                DateTime = DateTime.Now, TimeLimit = 50
            };

            var result = controller.Create(debate) as RedirectResult;

            mockRepo.Verify(d => d.AddDebate(debate), Times.Once);
            mockRepo.Verify(d => d.Save(), Times.Once);

            Assert.IsNotNull(result);
        }
    using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("RateMyDebate")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
```

```
[assembly: AssemblyProduct("RateMyDebate")]
[assembly: AssemblyCopyright("Copyright ©  2014")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components.  If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("61e46e63-42ce-4868-b2e9-e2e143e71993")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Revision and Build Numbers
// by using the '*' as shown below:
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]

using System.Web;
using System.Web.Optimization;

namespace RateMyDebate
{
    public class BundleConfig
    {
        // For more information on bundling, visit
http://go.microsoft.com/fwlink/?LinkId=301862
        public static void RegisterBundles(BundleCollection bundles)
        {
            bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                        "~/Scripts/jquery-{version}.js"));

            bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
                        "~/Scripts/jquery.validate*"));

            // Use the development version of Modernizr to develop with and learn from.
Then, when you're
            // ready for production, use the build tool at http://modernizr.com to pick only
the tests you need.
            bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                        "~/Scripts/modernizr-*"));

            bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
                      "~/Scripts/bootstrap.js",
                      "~/Scripts/respond.js"));

            bundles.Add(new StyleBundle("~/Content/css").Include(
                      "~/Content/bootstrap.css",
                      "~/Content/site.css"));
        }
    }
}
```

```
using System.Web;
using System.Web.Mvc;

namespace RateMyDebate
{
    public class FilterConfig
    {
        public static void RegisterGlobalFilters(GlobalFilterCollection filters)
        {
            filters.Add(new HandleErrorAttribute());
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace RateMyDebate
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "RasmusIndex", id =
UrlParameter.Optional }
            );
        }
    }
}

using Microsoft.AspNet.Identity;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Owin;

namespace RateMyDebate
{
    public partial class Startup
    {
        // For more information on configuring authentication, please visit
http://go.microsoft.com/fwlink/?LinkId=301864
        public void ConfigureAuth(IAppBuilder app)
        {
            // Enable the application to use a cookie to store information for the signed in
user
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login")
            });
```

```
            // Use a cookie to temporarily store information about a user logging in with a
third party login provider
            app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

            // Uncomment the following lines to enable logging in with third party login
providers
            //app.UseMicrosoftAccountAuthentication(
            //    clientId: "",
            //    clientSecret: "");

            //app.UseTwitterAuthentication(
            //   consumerKey: "",
            //   consumerSecret: "");

            //app.UseFacebookAuthentication(
            //   appId: "",
            //   appSecret: "");

            //app.UseGoogleAuthentication();
        }
    }
}
```