ADMIN CLASS

using System.ComponentModel.DataAnnotations;

[Display(Name = "Username")]

public string Admin\_Username { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Password")]

[StringLength(10, ErrorMessage = "The {0} must be at least {2} characters and max 10 characters long.", MinimumLength = 5)]

public string Admin\_Password { get; set; }

[Required]

[Display(Name = "Confirm Password")]

[DataType(DataType.Password)]

[Compare("Admin\_Password", ErrorMessage = "The password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

[Required]

[Display(Name = "Name")]

public string Admin\_Name { get; set; }

[Display(Name = "Middle Name")]

public string Admin\_MidName { get; set; }

[Required]

[Display(Name = "Surname")]

public string Admin\_Surname { get; set; }

[Required]

[Display(Name = "Date of Birth")]

[DataType(DataType.Date)]

public System.DateTime Admin\_DOB { get; set; }

[Required]

[Display(Name = "Gender")]

public string Admin\_Gender { get; set; }

[Required]

[Display(Name = "Email")]

[EmailAddress]

public string Admin\_Email { get; set; }

[Display(Name = "Recovery Question")]

public string Admin\_RecoveryQ { get; set; }

[Display(Name = "Recovery Answer")]

public string Admin\_RecoveryA { get; set; }

[Display(Name = "Image")]

public byte[] Admin\_Image { get; set; }

public string Admin\_ImagePath { get; set; }

CANDIDATE CLASS

[Display(Name = "ID")]

public string Candidate\_ID { get; set; }

[Required]

[Display(Name = "Name")]

public string Candidate\_Name { get; set; }

[Display(Name = "Middle Name")]

public string Candidate\_MidName { get; set; }

[Required]

[Display(Name = "Surname")]

public string Candidate\_Surname { get; set; }

[Required]

[Display(Name = "Student Number")]

public int Candidate\_StudentNumber { get; set; }

[Required]

[Display(Name = "Gender")]

public string Candidate\_Gender { get; set; }

[Required]

[Display(Name = "Date of Birth")]

[DataType(DataType.Date)]

public System.DateTime Candidate\_DOB { get; set; }

[Required]

[Display(Name = "Email")]

[EmailAddress]

public string Candidate\_Email { get; set; }

[Required]

[Display(Name = "Course")]

public string Candidate\_Course { get; set; }

[Required]

[Display(Name = "Year of Study")]

public int Candidate\_YearOfStudy { get; set; }

[Required]

[Display(Name = "Campus")]

public string Candidate\_Campus { get; set; }

[Required]

[Display(Name = "Position")]

public string Candidate\_Position { get; set; }

[Required]

[Display(Name = "Status")]

public string Candidate\_Status { get; set; }

[Display(Name = "Total Votes")]

public Nullable<int> Candidate\_TotalVotes { get; set; }

[Required]

[Display(Name = "Party")]

public int Party\_ID { get; set; }

//[Required]

[Display(Name = "Admin Incharge")]

public string Admin\_Username { get; set; }

[Display(Name = "Image")]

public byte[] Candidate\_Image { get; set; }

public string Candidate\_ImagePath { get; set; }

public bool IsChecked { get; set; }

public string GenerateCandID()

{

string x = Candidate\_Name.Substring(0, 1);

string y = Candidate\_StudentNumber.ToString();

string z = Candidate\_Surname.Substring(0, 1);

string cand = x + y + z;

return cand;

}

PARTY CLASS

[Required]

[Display(Name = "Party ID")]

public int Party\_ID { get; set; }

[Required]

[Display(Name = "Party Name")]

public string Party\_Name { get; set; }

[Required]

[Display(Name = "Party Description")]

public string Party\_Description { get; set; }

[Display(Name = "Number of Candidates")]

public Nullable<int> Party\_NoOfCandidates { get; set; }

[Required]

[Display(Name = "Status")]

public string Party\_Status { get; set; }

[Display(Name = "Total Votes")]

public Nullable<int> Party\_TotalVotes { get; set; }

//[Required]

[Display(Name = "Admin Incharge")]

public string Admin\_Username { get; set; }

[Display(Name = "Image")]

public byte[] Party\_Image { get; set; }

public string Party\_ImagePath { get; set; }

STUDENT CLASS

[Required]

[Display(Name = "Student Number")]

public int Student\_Number { get; set; }

[Required]

[Display(Name = "Password")]

[DataType(DataType.Password)]

public string Student\_Password { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Confirm Password")]

[Compare("Student\_Password", ErrorMessage = "The password and confirmation password do not match.")]

public string ConfirmPassword { get; set; }

[Required]

[Display(Name = "Email")]

[EmailAddress]

public string Student\_Email { get; set; }

[Required]

[Display(Name = "Gender")]

public string Student\_Gender { get; set; }

[Required]

[Display(Name = "Date of Birth")]

[DataType(DataType.Date)]

public System.DateTime Student\_DOB { get; set; }

[Required]

[Display(Name = "Course")]

public string Student\_Course { get; set; }

[Required]

[Display(Name = "Year of Study")]

public int Student\_YearOfStudy { get; set; }

[Required]

[Display(Name = "Campus")]

public string Student\_Campus { get; set; }

[Display(Name = "Voted")]

public string Student\_Voted { get; set; }

[Display(Name = "Recovery Question")]

public string Student\_RecoveryQ { get; set; }

[Display(Name = "Recovery Answer")]

public string Student\_RecoveryA { get; set; }

[Display(Name = "Image")]

public byte[] Student\_Image { get; set; }

public string Student\_ImagePath { get; set; }

VOTE CLASS

[Display(Name = "Vote ID")]

public int Vote\_ID { get; set; }

[Required]

[Display(Name = "Party")]

public int Party\_ID { get; set; }

[Required]

[Display(Name = "Candidate")]

public string Candidate\_ID { get; set; }

[Required]

[Display(Name = "Student Number")]

public int Student\_Number { get; set; }

[Display(Name = "Voting Date")]

[DataType(DataType.Date)]

public Nullable<System.DateTime> Vote\_Date { get; set; }

public List<Candidate> Candidates;

public virtual Candidate Candidate { get; set; }

public virtual Student Student { get; set; }

public virtual Party Party { get; set; }

VOTETIME CLASS

public int ID { get; set; }

[Required]

[Display(Name = "Start Date")]

[DataType(DataType.Date)]

public Nullable<System.DateTime> Start\_Date { get; set; }

[Required]

[Display(Name = "End Date")]

[DataType(DataType.Date)]

public Nullable<System.DateTime> End\_Date { get; set; }

LOGIN CONTROLLER

[HttpPost]

public ActionResult AdminLogin(Admin admin)

{

using (ELectDBEntities db = new ELectDBEntities())

{

var username = db.Admins.Where(x => x.Admin\_Username == admin.Admin\_Username).FirstOrDefault();

var password = db.Admins.Where(x => x.Admin\_Password == admin.Admin\_Password).FirstOrDefault();

if (username == null && password == null)

{

ViewBag.ErrorMessage = "Incorrect Username and Password";

return View("AdminLogin", admin);

}

else if (username == null)

{

ViewBag.ErrorMessage = "Incorrect Username";

return View("AdminLogin", admin);

}

else if (password == null)

{

ViewBag.ErrorMessage = "Incorrect Password";

return View("AdminLogin", admin);

}

else

{

Session["Admin\_Username"] = admin.Admin\_Username;

return RedirectToAction("AdminHome", "Home");

}

}

}

public ActionResult AdminLogOff()

{

string username = (string)Session["Admin\_Username"];

Session.Abandon();

return RedirectToAction("AdminLogin", "Login");

}

[HttpPost]

public ActionResult StudentLogin(Student student)

{

using (ELectDBEntities db = new ELectDBEntities())

{

var studentNum = db.Students.Where(x => x.Student\_Number == student.Student\_Number).FirstOrDefault();

var password = db.Students.Where(x => x.Student\_Password == student.Student\_Password).FirstOrDefault();

if (studentNum == null && password == null)

{

ViewBag.ErrorMessage = "Incorrect Username and Password";

return View("StudentLogin", student);

}

else if (studentNum == null)

{

ViewBag.ErrorMessage = "Incorrect Username";

return View("StudentLogin", student);

}

else if (password == null)

{

ViewBag.ErrorMessage = "Incorrect Password";

return View("StudentLogin", student);

}

else

{

Session["Student\_Number"] = student.Student\_Number;

return RedirectToAction("StudentHome", "Home");

}

}

}

public ActionResult StudentLogOff()

{

int studentNum = (int)Session["Student\_Number"];

Session.Abandon();

return RedirectToAction("StudentLogin", "Login");

}

}

ADMIN CONTROLLER

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Admin\_Username,Admin\_Password,ConfirmPassword,Admin\_Name,Admin\_MidName,Admin\_Surname,Admin\_DOB,Admin\_Gender,Admin\_Email,Admin\_RecoveryQ,Admin\_RecoveryA,Admin\_Image,Admin\_ImagePath")] Admin admin)

{

if (ModelState.IsValid)

{

if (db.Admins.Any(x => x.Admin\_Email == admin.Admin\_Email) || db.Admins.Any(x => x.Admin\_Username == admin.Admin\_Username))

{

ViewBag.DuplicateMessage = "This Admin username already exists.";

return View("Create", admin);

}

db.Admins.Add(admin);

db.SaveChanges();

ModelState.Clear();

ViewBag.SuccessMessage = "Registration Successful.";

return View("Create", new Admin());

}

return View(admin);

}

CANDIDATE CONTROLLER

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Candidate\_ID,Candidate\_Name,Candidate\_MidName,Candidate\_Surname,Candidate\_StudentNumber,Candidate\_Gender,Candidate\_DOB,Candidate\_Email,Candidate\_Course,Candidate\_YearOfStudy,Candidate\_Campus,Candidate\_Position,Candidate\_Status,Candidate\_TotalVotes,Party\_ID,Admin\_Username,Candidate\_Image,Candidate\_ImagePath")] Candidate candidate)

{

if (ModelState.IsValid)

{

candidate.Candidate\_ID = candidate.GenerateCandID();

candidate.Candidate\_TotalVotes = 0;

var admin = (string)Session["Admin\_Username"];

candidate.Admin\_Username = admin;

if (db.Candidates.Any(x => x.Candidate\_ID == candidate.Candidate\_ID) || db.Candidates.Any(x => x.Candidate\_StudentNumber == candidate.Candidate\_StudentNumber))

{

ModelState.Clear();

ViewBag.DuplicateMessage = "This candidate already exists.";

return View("Create");

}

var studentNum = db.MockStudents.Find(candidate.Candidate\_StudentNumber);

if (studentNum == null)

{

ModelState.Clear();

ViewBag.ErrorMessage = "Invalid candidate! Not a registered DUT student!";

return View("Create", candidate);

}

var party = db.Parties.Find(candidate.Party\_ID);

if (party != null)

{

party.Party\_NoOfCandidates += 1;

db.Entry(party).State = EntityState.Modified;

db.SaveChanges();

}

else if (party == null)

{

ModelState.Clear();

ViewBag.ErrorMessage = "Incorrect Party!";

return View("Create", candidate);

}

db.Candidates.Add(candidate);

db.SaveChanges();

ModelState.Clear();

ViewBag.SuccessMessage = "Registration Successfully.";

return View("Create", new Candidate());

}

ViewBag.Admin\_Username = new SelectList(db.Admins, "Admin\_Username", "Admin\_Password", candidate.Admin\_Username);

ViewBag.Party\_ID = new SelectList(db.Parties, "Party\_ID", "Party\_Name", candidate.Party\_ID);

return View(candidate);

}

PARTY CONTROLLER

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Party\_ID,Party\_Name,Party\_Description,Party\_NoOfCandidates,Party\_Status,Party\_TotalVotes,Admin\_Username,Party\_Image,Party\_ImagePath")] Party party)

{

if (ModelState.IsValid)

{

var admin = (string)Session["Admin\_Username"];

party.Admin\_Username = admin;

party.Party\_NoOfCandidates = 0;

if (db.Parties.Any(x => x.Party\_Name == party.Party\_Name) || db.Parties.Any(x => x.Party\_Description == party.Party\_Description))

{

ModelState.Clear();

ViewBag.DuplicateMessage = "Party already exists.";

return View("Create", party);

}

db.Parties.Add(party);

db.SaveChanges();

return RedirectToAction("Index");

}

ViewBag.Admin\_Username = new SelectList(db.Admins, "Admin\_Username", "Admin\_Password", party.Admin\_Username);

return View(party);

}

STUDENT CONTROLLER

public ActionResult Details(int? id)

{

id = (int)Session["Student\_Number"];

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Student student = db.Students.Find(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Student\_Number,Student\_Password,ConfirmPassword,Student\_Email,Student\_Gender,Student\_DOB,Student\_Course,Student\_YearOfStudy,Student\_Campus,Student\_Voted,Student\_RecoveryQ,Student\_RecoveryA,Student\_Image,Student\_ImagePath")] Student student)

{

if (ModelState.IsValid)

{

student.Student\_Voted = "False";

if (db.Students.Any(x => x.Student\_Number == student.Student\_Number))

{

ModelState.Clear();

ViewBag.DuplicateMessage = "You are already registered.";

return View("Create", student);

}

var studentNum = db.MockStudents.Find(student.Student\_Number);

if (studentNum == null)

{

ModelState.Clear();

ViewBag.ErrorMessage = "Invalid student! You are not a registered DUT student!";

return View("Create", student);

}

//student.Student\_Recovery = " ";

db.Students.Add(student);

db.SaveChanges();

ModelState.Clear();

ViewBag.SuccessMessage = "Registration Successful.";

return View("Create", new Student());

//return RedirectToAction("Index");

}

return View(student);

}

public ActionResult Edit(int? id)

{

id = (int)Session["Student\_Number"];

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Student student = db.Students.Find(id);

if (student == null)

{

return HttpNotFound();

}

return View(student);

}

VOTE CONTROLLER

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult EditVoteTime([Bind(Include = "ID,Start\_Date,End\_Date")] VoteTime voteTime)

{

if (ModelState.IsValid)

{

db.Entry(voteTime).State = EntityState.Modified;

db.SaveChanges();

ViewBag.SuccessMessage = "Voting Time changed successfully!";

return RedirectToAction("EditVoteTime");

}

return View(voteTime);

}

public ActionResult Create()

{

var time = db.VoteTimes.Find(1);

if ((time.Start\_Date <= DateTime.Now.Date) && DateTime.Now.Date < time.End\_Date)

{

RedirectToAction("Create");

}

else if (DateTime.Now.Date > time.Start\_Date)

{

return RedirectToAction("Construct");

}

else if (time.End\_Date == DateTime.Now.Date)

{

return RedirectToAction("Construct");

}

else

{

return RedirectToAction("Construct");

}

ViewBag.Candidate\_ID = new SelectList(db.Candidates, "Candidate\_ID", "Candidate\_Name");

ViewBag.Party\_ID = new SelectList(db.Parties, "Party\_ID", "Party\_Name");

ViewBag.Student\_Number = new SelectList(db.Students, "Student\_Number", "Student\_Password");

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Vote\_ID,Party\_ID,Candidate\_ID,Student\_Number")] Vote vote)

{

if (ModelState.IsValid)

{

var studentNum = (int)Session["Student\_Number"];

vote.Student\_Number = studentNum;

if (db.Votes.Any(x => x.Student\_Number == vote.Student\_Number))

{

ModelState.Clear();

ViewBag.DuplicateMessage = "You have already voted!";

return View("Create", vote);

}

ELectDBEntities si = new ELectDBEntities();

Candidate cand = si.Candidates.Find(vote.Candidate\_ID);

if (cand != null)

{

cand.Candidate\_TotalVotes += 1;

si.Entry(cand).State = EntityState.Modified;

si.SaveChanges();

}

else if (cand == null)

{

ModelState.Clear();

ViewBag.ErrorMessage = "Incorrect Candidate!";

return View("Create", vote);

}

var partyID = cand.Party\_ID;

vote.Party\_ID = partyID;

ELectDBEntities se = new ELectDBEntities();

var party = se.Parties.Find(vote.Party\_ID);

if (party != null)

{

party.Party\_TotalVotes += 1;

se.Entry(party).State = EntityState.Modified;

se.SaveChanges();

}

db.Votes.Add(vote);

db.SaveChanges();

ModelState.Clear();

ViewBag.SuccessMessage = "Voted Successfully.";

return View("Create", new Vote());

}

ViewBag.Candidate\_ID = new SelectList(db.Candidates, "Candidate\_ID", "Candidate\_Name", vote.Candidate\_ID);

ViewBag.Party\_ID = new SelectList(db.Parties, "Party\_ID", "Party\_Name", vote.Party\_ID);

ViewBag.Student\_Number = new SelectList(db.Students, "Student\_Number", "Student\_Password", vote.Student\_Number);

return View(vote);

}