using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using ICSharpCode.SharpZipLib.Zip;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.EntityFrameworkCore;

using UNFHackAThon.Data;

using UNFHackAThon.Models;

using UNFHackAThon.Models.ViewModels;

using UNFHackAThon.Utility;

namespace UNFHackAThon.Controllers

{

[Authorize()]

[Area("Participants")]

public class HomeController : Controller

{

private readonly ApplicationDbContext \_db;

private IHostingEnvironment Environment;

public HomeController(ApplicationDbContext db, IHostingEnvironment \_environment)

{

\_db = db;

Environment = \_environment;

}

public async Task<IActionResult> Index()

{

IndexViewModel IndexVM = new IndexViewModel()

{

CompetitionItem = await \_db.CompetitionItem.Include(m => m.Competition).Include(m => m.SubCompetition).ToListAsync(),

Competition = await \_db.Competition.ToListAsync(),

};

var claimsIdentity = (ClaimsIdentity)User.Identity;

var claim = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier);

if (claim != null)

{

var cnt = \_db.CompetitionCart.Where(u => u.ApplicationUserId == claim.Value).ToList().Count();

HttpContext.Session.SetInt32("ssCartCount", cnt);

}

return View(IndexVM);

}

[Authorize]

public async Task<IActionResult> Details(int id)

{

var competitionItemFromDb = await \_db.CompetitionItem.Include(m => m.Competition).Include(m => m.SubCompetition).Where(m => m.Id == id).FirstOrDefaultAsync();

CompetitionCart cartObj = new CompetitionCart()

{

CompetitionItem = competitionItemFromDb,

CompetitionItemId = competitionItemFromDb.Id

};

return View(cartObj);

}

[Authorize]

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Details(CompetitionCart CartObject)

{

CartObject.Id = 0;

if (ModelState.IsValid)

{

var claimsIdentity = (ClaimsIdentity)this.User.Identity;

var claim = claimsIdentity.FindFirst(ClaimTypes.NameIdentifier);

CartObject.ApplicationUserId = claim.Value;

CompetitionCart cartFromDb = await \_db.CompetitionCart.Where(c => c.ApplicationUserId == CartObject.ApplicationUserId

&& c.CompetitionItemId == CartObject.CompetitionItemId).FirstOrDefaultAsync();

if (cartFromDb == null)

{

await \_db.CompetitionCart.AddAsync(CartObject);

}

else

{

cartFromDb.Count = cartFromDb.Count + CartObject.Count;

}

await \_db.SaveChangesAsync();

var count = \_db.CompetitionCart.Where(c => c.ApplicationUserId == CartObject.ApplicationUserId).ToList().Count();

HttpContext.Session.SetInt32("ssCartCount", count);

return RedirectToAction("Index");

}

else

{

var competitionItemFromDb = await \_db.CompetitionItem.Include(m => m.Competition).Include(m => m.SubCompetition).Where(m => m.Id == CartObject.CompetitionItemId).FirstOrDefaultAsync();

CompetitionCart cartObj = new CompetitionCart()

{

CompetitionItem = competitionItemFromDb,

CompetitionItemId = competitionItemFromDb.Id

};

return View(cartObj);

}

}

public ActionResult Code()

{

return View();

}

[HttpPost]

// /Participants/Home/Code/3

public IActionResult Code(int competitionId, [FromBody] List<IFormFile> postedFiles)

{

string wwwPath = this.Environment.WebRootPath;

string contentPath = this.Environment.ContentRootPath;

// var userId = get userId through this.User....

// basePath/{competitionId}/{userId}/\*.\*

// basePath/{userId}/{competitionId}/\*.\*

// basePath/user/{userId}/competition/{competitionId}/\*.\*

// need to create a storage path that is unique per user + competition combiantion

//string path = Path.Combine(this.Environment.WebRootPath, "Code");

var userId = 12;

var path = Path.Combine(this.Environment.WebRootPath, "user", userId.ToString(), "competition", competitionId.ToString());

if (!Directory.Exists(path))

{

Directory.CreateDirectory(path);

}

List<string> uploadedFiles = new List<string>();

foreach (IFormFile postedFile in postedFiles)

{

string fileName = Path.GetFileName(postedFile.FileName);

using (FileStream stream = new FileStream(Path.Combine(path, fileName), FileMode.Create))

{

postedFile.CopyTo(stream);

uploadedFiles.Add(fileName);

ViewBag.Message += string.Format("<b>{0}</b> uploaded.<br />", fileName);

}

}

return View(uploadedFiles);

}

public FileResult DownloadZipFile(/\*competitionId, userId \*/)

{

var fileName = string.Format("{0}\_CodeFiles.zip", DateTime.Today.Date.ToString("dd-MM-yyyy") + "\_1");

//

string wwwPath = this.Environment.WebRootPath;

string contentPath = this.Environment.ContentRootPath;

// need to create a storage path that is unique per user + competition combiantion

string path = Path.Combine(this.Environment.WebRootPath, "Code");

var tempOutPutPath = path + fileName;

using (ZipOutputStream s = new ZipOutputStream(System.IO.File.Create(tempOutPutPath)))

{

s.SetLevel(9); // 0-9, 9 being the highest compression

byte[] buffer = new byte[4096];

var ImageList = new List<string>();

ImageList.Add(wwwPath + ("/Code/ArrayPractice.java"));

//for (int i = 0; i < ImageList.Count; i++)

foreach (var file in Directory.EnumerateFiles(path))

{

ZipEntry entry = new ZipEntry(file);

entry.DateTime = DateTime.Now;

entry.IsUnicodeText = true;

s.PutNextEntry(entry);

using (FileStream fs = System.IO.File.OpenRead(file))

{

int sourceBytes;

do

{

sourceBytes = fs.Read(buffer, 0, buffer.Length);

s.Write(buffer, 0, sourceBytes);

} while (sourceBytes > 0);

}

}

s.Finish();

s.Flush();

s.Close();

}

byte[] finalResult = System.IO.File.ReadAllBytes(tempOutPutPath);

if (System.IO.File.Exists(tempOutPutPath))

System.IO.File.Delete(tempOutPutPath);

if (finalResult == null || !finalResult.Any())

throw new Exception(String.Format("No Files found with Code"));

return File(finalResult, "application/zip", fileName);

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}