namespace EX.Migrations

{

using EX.Models;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Data.Entity.Migrations;

using System.Linq;

internal sealed class DbMigrationsConfig : DbMigrationsConfiguration<EX.Models.ApplicationDbContext>

{

public DbMigrationsConfig()

{

AutomaticMigrationsEnabled = true;

AutomaticMigrationDataLossAllowed = true;

}

protected override void Seed(ApplicationDbContext context)

{

if (!context.Users.Any())

{

// ADMIN user

var adminEmail = "admin@fpt.com";

var adminUserName = adminEmail;

var adminPassword = "1234";

string adminRole = "Admin";

string adminDepartment = "Administrator";

CreateAdmin(context, adminEmail, adminUserName, adminPassword, adminRole, adminDepartment);

//QA Manager user

var QA\_M\_Email = "QA\_Manager@fpt.com";

var QA\_M\_UserName = QA\_M\_Email;

var QA\_M\_Password = "1234";

string[] QA\_M\_Role = { "Staff", "QA Manager" };

string QA\_M\_Department = "QA";

CreateQA\_Manager(context, QA\_M\_Email, QA\_M\_UserName, QA\_M\_Password, QA\_M\_Role, QA\_M\_Department);

//QA Coordinator IT user

var QA\_IT\_Email = "QA\_IT@fpt.com";

var QA\_IT\_UserName = QA\_IT\_Email;

var QA\_IT\_Password = "1234";

string[] QA\_IT\_Role = { "Staff", "QA\_C" };

string QA\_IT\_Department = "IT";

CreateQA\_IT(context, QA\_IT\_Email, QA\_IT\_UserName, QA\_IT\_Password, QA\_IT\_Role, QA\_IT\_Department);

//QA Coordinator HR user

var QA\_HR\_Email = "QA\_HR@fpt.com";

var QA\_HR\_UserName = QA\_HR\_Email;

var QA\_HR\_Password = "1234";

string[] QA\_HR\_Role = { "Staff", "QA\_C" };

string QA\_HR\_Department = "HR";

CreateQA\_HR(context, QA\_HR\_Email, QA\_HR\_UserName, QA\_HR\_Password, QA\_HR\_Role, QA\_HR\_Department);

//Staff in QA department user

var Staff\_QA\_Email = "Staff\_QA@fpt.com";

var Staff\_QA\_UserName = Staff\_QA\_Email;

var Staff\_QA\_Password = "1234";

string[] Staff\_QA\_Role = { "Staff" };

string Staff\_QA\_Department = "QA";

Create\_Staff\_QA(context, Staff\_QA\_Email, Staff\_QA\_UserName, Staff\_QA\_Password, Staff\_QA\_Role, Staff\_QA\_Department);

//staff in HR department user

var Staff\_HR\_Email = "Staff\_HR@fpt.com";

var Staff\_HR\_UserName = Staff\_HR\_Email;

var Staff\_HR\_Password = "1234";

string[] Staff\_HR\_Role = { "Staff" };

string Staff\_HR\_Department = "HR";

Create\_Staff\_HR(context, Staff\_HR\_Email, Staff\_HR\_UserName, Staff\_HR\_Password, Staff\_HR\_Role, Staff\_HR\_Department);

//staff in IT department user

var Staff\_IT\_Email = "Staff\_IT@fpt.com";

var Staff\_IT\_UserName = Staff\_IT\_Email;

var Staff\_IT\_Password = "1234";

string[] Staff\_IT\_Role = { "Staff" };

string Staff\_IT\_Department = "IT";

Create\_Staff\_IT(context, Staff\_IT\_Email, Staff\_IT\_UserName, Staff\_IT\_Password, Staff\_IT\_Role, Staff\_IT\_Department);

CreateSeveralCategory(context);

CreateSeveralEvents(context);

}

}

private void CreateSeveralCategory(ApplicationDbContext context)

{

context.Categories.Add(new Category()

{

Category\_Name = "Sport",

});

context.Categories.Add(new Category()

{

Category\_Name = "Holiday",

});

context.Categories.Add(new Category()

{

Category\_Name = "Public Event",

});

}

private void CreateSeveralEvents(ApplicationDbContext context)

{

context.Ideas.Add(new Idea()

{

Content = "1st Idea",

CateId = 1,

Date = DateTime.Now.AddDays(5).AddHours(21).AddMinutes(30),

Author = context.Users.First(),

Author\_Email = context.Users.First().Email

});

context.Ideas.Add(new Idea()

{

Content = "2nd Idea",

CateId = 2,

Date = DateTime.Now.AddDays(2).AddHours(10).AddMinutes(30),

IsAnonymous = true,

Author = context.Users.First(),

Author\_Email = context.Users.First().Email,

Comments = new HashSet<Comment>()

{

new Comment() {Text = "<Anonymous> comment", Author = context.Users.First(), IsAnonymous = true, AuthorName = context.Users.First().Email },

new Comment() {Text = "UserComment", Author = context.Users.First(), IsAnonymous = false, AuthorName = context.Users.First().Email}

}

});

}

private void CreateAdmin(ApplicationDbContext context, string adminEmail, string adminUserName, string adminPassword, string adminRole, string adminDepartment)

{

var adminUser = new ApplicationUser

{

UserName = adminUserName,

Email = adminEmail,

Department = adminDepartment

};

var userStore = new UserStore<ApplicationUser>(context);

var userManager = new UserManager<ApplicationUser>(userStore);

userManager.PasswordValidator = new PasswordValidator

{

RequiredLength = 4,

RequireNonLetterOrDigit = false,

RequireDigit = false,

RequireLowercase = false,

RequireUppercase = false,

};

var userCreateResult = userManager.Create(adminUser, adminPassword);

if (!userCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", userCreateResult.Errors));

}

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

var roleCreateResult = roleManager.Create(new IdentityRole(adminRole));

if (!roleCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", roleCreateResult.Errors));

}

var addTrainingstaffRoleResult = userManager.AddToRole(adminUser.Id, adminRole);

if (!addTrainingstaffRoleResult.Succeeded)

{

throw new Exception(string.Join("; ", addTrainingstaffRoleResult.Errors));

}

}

private void CreateQA\_Manager(ApplicationDbContext context, string QA\_M\_Email, string QA\_M\_UserName, string QA\_M\_Password, string[] QA\_M\_Role, string QA\_M\_Department)

{

var adminUser = new ApplicationUser

{

UserName = QA\_M\_UserName,

Email = QA\_M\_Email,

Department = QA\_M\_Department

};

var userStore = new UserStore<ApplicationUser>(context);

var userManager = new UserManager<ApplicationUser>(userStore);

userManager.PasswordValidator = new PasswordValidator

{

RequiredLength = 4,

RequireNonLetterOrDigit = false,

RequireDigit = false,

RequireLowercase = false,

RequireUppercase = false,

};

var userCreateResult = userManager.Create(adminUser, QA\_M\_Password);

if (!userCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", userCreateResult.Errors));

}

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

foreach (var roleName in QA\_M\_Role)

{

var roleExist = roleManager.RoleExists(roleName);

if (!roleExist)

{

var role = new IdentityRole();

role.Name = roleName;

var roleCreateResult = roleManager.Create(new IdentityRole(roleName));

if (!roleCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", roleCreateResult.Errors));

}

var addTrainingstaffRoleResult = userManager.AddToRole(adminUser.Id, roleName);

if (!addTrainingstaffRoleResult.Succeeded)

{

throw new Exception(string.Join("; ", addTrainingstaffRoleResult.Errors));

}

}

}

}

private void CreateQA\_IT(ApplicationDbContext context, string QA\_M\_Email, string QA\_M\_UserName, string QA\_M\_Password, string[] QA\_M\_Role, string QA\_M\_Department)

{

var adminUser = new ApplicationUser

{

UserName = QA\_M\_UserName,

Email = QA\_M\_Email,

Department = QA\_M\_Department

};

var userStore = new UserStore<ApplicationUser>(context);

var userManager = new UserManager<ApplicationUser>(userStore);

userManager.PasswordValidator = new PasswordValidator

{

RequiredLength = 4,

RequireNonLetterOrDigit = false,

RequireDigit = false,

RequireLowercase = false,

RequireUppercase = false,

};

var userCreateResult = userManager.Create(adminUser, QA\_M\_Password);

if (!userCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", userCreateResult.Errors));

}

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

foreach (var roleName in QA\_M\_Role)

{

var roleExist = roleManager.RoleExists(roleName);

if (!roleExist)

{

var role = new IdentityRole();

role.Name = roleName;

var roleCreateResult = roleManager.Create(new IdentityRole(roleName));

if (!roleCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", roleCreateResult.Errors));

}

var addTrainingstaffRoleResult = userManager.AddToRole(adminUser.Id, roleName);

if (!addTrainingstaffRoleResult.Succeeded)

{

throw new Exception(string.Join("; ", addTrainingstaffRoleResult.Errors));

}

}

}

}

private void CreateQA\_HR(ApplicationDbContext context, string QA\_M\_Email, string QA\_M\_UserName, string QA\_M\_Password, string[] QA\_M\_Role, string QA\_M\_Department)

{

var adminUser = new ApplicationUser

{

UserName = QA\_M\_UserName,

Email = QA\_M\_Email,

Department = QA\_M\_Department

};

var userStore = new UserStore<ApplicationUser>(context);

var userManager = new UserManager<ApplicationUser>(userStore);

userManager.PasswordValidator = new PasswordValidator

{

RequiredLength = 4,

RequireNonLetterOrDigit = false,

RequireDigit = false,

RequireLowercase = false,

RequireUppercase = false,

};

var userCreateResult = userManager.Create(adminUser, QA\_M\_Password);

if (!userCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", userCreateResult.Errors));

}

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

foreach (var roleName in QA\_M\_Role)

{

var roleExist = roleManager.RoleExists(roleName);

if (!roleExist)

{

var role = new IdentityRole();

role.Name = roleName;

var roleCreateResult = roleManager.Create(new IdentityRole(roleName));

if (!roleCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", roleCreateResult.Errors));

}

var addTrainingstaffRoleResult = userManager.AddToRole(adminUser.Id, roleName);

if (!addTrainingstaffRoleResult.Succeeded)

{

throw new Exception(string.Join("; ", addTrainingstaffRoleResult.Errors));

}

}

}

}

private void Create\_Staff\_QA(ApplicationDbContext context, string QA\_M\_Email, string QA\_M\_UserName, string QA\_M\_Password, string[] QA\_M\_Role, string QA\_M\_Department)

{

var adminUser = new ApplicationUser

{

UserName = QA\_M\_UserName,

Email = QA\_M\_Email,

Department = QA\_M\_Department

};

var userStore = new UserStore<ApplicationUser>(context);

var userManager = new UserManager<ApplicationUser>(userStore);

userManager.PasswordValidator = new PasswordValidator

{

RequiredLength = 4,

RequireNonLetterOrDigit = false,

RequireDigit = false,

RequireLowercase = false,

RequireUppercase = false,

};

var userCreateResult = userManager.Create(adminUser, QA\_M\_Password);

if (!userCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", userCreateResult.Errors));

}

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

foreach (var roleName in QA\_M\_Role)

{

var roleExist = roleManager.RoleExists(roleName);

if (!roleExist)

{

var role = new IdentityRole();

role.Name = roleName;

var roleCreateResult = roleManager.Create(new IdentityRole(roleName));

if (!roleCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", roleCreateResult.Errors));

}

var addTrainingstaffRoleResult = userManager.AddToRole(adminUser.Id, roleName);

if (!addTrainingstaffRoleResult.Succeeded)

{

throw new Exception(string.Join("; ", addTrainingstaffRoleResult.Errors));

}

}

}

}

private void Create\_Staff\_HR(ApplicationDbContext context, string QA\_M\_Email, string QA\_M\_UserName, string QA\_M\_Password, string[] QA\_M\_Role, string QA\_M\_Department)

{

var adminUser = new ApplicationUser

{

UserName = QA\_M\_UserName,

Email = QA\_M\_Email,

Department = QA\_M\_Department

};

var userStore = new UserStore<ApplicationUser>(context);

var userManager = new UserManager<ApplicationUser>(userStore);

userManager.PasswordValidator = new PasswordValidator

{

RequiredLength = 4,

RequireNonLetterOrDigit = false,

RequireDigit = false,

RequireLowercase = false,

RequireUppercase = false,

};

var userCreateResult = userManager.Create(adminUser, QA\_M\_Password);

if (!userCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", userCreateResult.Errors));

}

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

foreach (var roleName in QA\_M\_Role)

{

var roleExist = roleManager.RoleExists(roleName);

if (!roleExist)

{

var role = new IdentityRole();

role.Name = roleName;

var roleCreateResult = roleManager.Create(new IdentityRole(roleName));

if (!roleCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", roleCreateResult.Errors));

}

var addTrainingstaffRoleResult = userManager.AddToRole(adminUser.Id, roleName);

if (!addTrainingstaffRoleResult.Succeeded)

{

throw new Exception(string.Join("; ", addTrainingstaffRoleResult.Errors));

}

}

}

}

private void Create\_Staff\_IT(ApplicationDbContext context, string QA\_M\_Email, string QA\_M\_UserName, string QA\_M\_Password, string[] QA\_M\_Role, string QA\_M\_Department)

{

var adminUser = new ApplicationUser

{

UserName = QA\_M\_UserName,

Email = QA\_M\_Email,

Department = QA\_M\_Department

};

var userStore = new UserStore<ApplicationUser>(context);

var userManager = new UserManager<ApplicationUser>(userStore);

userManager.PasswordValidator = new PasswordValidator

{

RequiredLength = 4,

RequireNonLetterOrDigit = false,

RequireDigit = false,

RequireLowercase = false,

RequireUppercase = false,

};

var userCreateResult = userManager.Create(adminUser, QA\_M\_Password);

if (!userCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", userCreateResult.Errors));

}

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(context));

foreach (var roleName in QA\_M\_Role)

{

var roleExist = roleManager.RoleExists(roleName);

if (!roleExist)

{

var role = new IdentityRole();

role.Name = roleName;

var roleCreateResult = roleManager.Create(new IdentityRole(roleName));

if (!roleCreateResult.Succeeded)

{

throw new Exception(string.Join("; ", roleCreateResult.Errors));

}

var addTrainingstaffRoleResult = userManager.AddToRole(adminUser.Id, roleName);

if (!addTrainingstaffRoleResult.Succeeded)

{

throw new Exception(string.Join("; ", addTrainingstaffRoleResult.Errors));

}

}

}

}

}

}