DOTNET CODE SNIPPETS

UserController with update password and send email

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Mail;
using System.Security.Cryptography;
using System.Threading.Tasks;
using KTU_Backend.DAL;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using StudentAttachmentV1.DAL;

namespace KTU_Backend.Controllers
{
[ApiController]
[Route("api/Auth")]


public class UserController : ControllerBase
{

private readonly StudentAttachmentContext _context;
public UserController(StudentAttachmentContext context)
{
_context = context;
}



[HttpPost("forgot")]
public async Task<IActionResult>ForgotPassword(string email)
{

var user = await _context.Accounts.FirstOrDefaultAsync(u => u.Email == email);
if(user==null){
return BadRequest("User " + email + " does not exist");
}

user.PasswordResetToken = CreateRandomToken();
user.ResetTokenExpires = DateTime.Now.AddDays(1);
await _context.SaveChangesAsync();

try
{
await SendPasswordResetEmail(email, user.PasswordResetToken);
return Ok("You may now reset your password");
}
```

```
catch (Exception ex)
{
// Handle exception if there was an error sending the email
// You can log the error or return an appropriate error message
return BadRequest("Failed to send password reset email. Please try again later.");
}

}


private async Task SendPasswordResetEmail(string email, string token)
{
string subject = "Password Reset";
string body = $"Dear {email},\n\n" +
"You have requested to reset your password. Please use the following token:\n\n" +
$"{token}\n\n" +
"This token will expire in 24 hours.\n\n" +
"Regards,\n" +
"Glydetek Group";

using (SmtpClient smtpClient = new SmtpClient("us2.smtp.mailhostbox.com", 25))
{
smtpClient.EnableSsl = true;
smtpClient.UseDefaultCredentials = false;
smtpClient.Credentials = new NetworkCredential("support@glydetek.com",
"eSM)TxT)0");

MailMessage mailMessage = new MailMessage();
mailMessage.From = new MailAddress("support@glydetek.com");
mailMessage.To.Add(email);
mailMessage.Subject = subject;
mailMessage.Body = body;

await smtpClient.SendMailAsync(mailMessage);
}
}




[HttpPost("reset-password")]
public async Task<IActionResult>ResetPassword(ResetPasswordRequest request)
{


//The PasswordResetToken which i declared in the User.cs model and it is made
availabe in the
//DataContext.cs through the DbSet shouuld be equal to the Token the user will
declare in the ResetPasswordRequest model
```

```
var user = await _context.Accounts.FirstOrDefaultAsync(u => u.PasswordResetToken
== request.Token);
if(user==null || user.ResetTokenExpires<DateTime.Now ){
return BadRequest("Invalid Token");
}
user.Password = BCrypt.Net.BCrypt.HashPassword(request.Password);
user.PasswordResetToken = null;
user.ResetTokenExpires = null;

await _context.SaveChangesAsync();

return Ok("Password successfully reset.");

}



[HttpPost("login")]
public async Task<IActionResult> Login(UserLoginHydot request)
{
var user = await _context.Accounts.FirstOrDefaultAsync(u => u.Email ==
request.Email);
if (user == null)
{
return BadRequest("User not found");
}

if (string.IsNullOrEmpty(user.Password))
{
return BadRequest("User password not set");
}

if (!BCrypt.Net.BCrypt.Verify(request.Password, user.Password))
{
return BadRequest("Incorrect password");
}

return Ok($"Welcome Back, {user.Name}!");
}




private static void CreatePasswordHash(string password, out byte[] passwordHash,
out byte[] passwordSalt)
{
using (var hmac = new HMACSHA512())
{
passwordSalt = hmac.Key;
passwordHash = hmac.ComputeHash(System.Text.Encoding.UTF8.GetBytes(password));
```

```
}

}

private string CreateRandomToken(){
return Convert.ToHexString(RandomNumberGenerator.GetBytes(64));
}


}
}
```

Upload Images And Files

```
using System;
using System.IO;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace HyStudy.Controllers
{
[ApiController] // Indicates that this class is an API controller
[Route("api/[controller]")] // Specifies the route prefix for this controller
public class ImageUploadController : ControllerBase
{
private readonly IWebHostEnvironment _environment;

public ImageUploadController(IWebHostEnvironment environment)
{
_environment = environment;
}

// Model representing the uploaded file
public class FileUploadApi
{
public IFormFile files { get; set; }
}

// HTTP POST endpoint for file upload
[HttpPost]
[Consumes("multipart/form-data")] // Specify the supported media type as form data
public async Task<IActionResult> Post([FromForm] FileUploadApi objFile)
{
try
{
// Check if a file was uploaded
if (objFile?.files == null || objFile.files.Length <= 0)
{
```

```
return BadRequest("No file uploaded."); // Return a 400 Bad Request if no file is
uploaded
}

// Create the "Upload" folder under the wwwroot folder
var uploadFolder = Path.Combine(_environment.WebRootPath, "Upload");

if (!Directory.Exists(uploadFolder))
{
Directory.CreateDirectory(uploadFolder);
}

// Generate a unique filename for the uploaded file
var uniqueFileName = Guid.NewGuid().ToString() + "<=> [" +
objFile.files.FileName+"]";

// Save the uploaded file to the "Upload" folder
using (FileStream fileStream = new FileStream(Path.Combine(uploadFolder,
uniqueFileName), FileMode.Create))
{
await objFile.files.CopyToAsync(fileStream);
fileStream.Flush();
}

return Ok("Upload successful: " + uniqueFileName); // Return a 200 OK response
with the unique filename
}
catch (Exception ex)
{
return StatusCode(500, ex.Message); // Return a 500 Internal Server Error for any
other exception
}
}
}
}
```