Studio 7 & 8
Name and Student Id: Nicolas Pallant 28785959

Self-Evaluation:

Need Help	Work in Progress	Pass	Credit	Distinction	High Distinction

TASK 7.1

GROUP MEMBERS

- Nicolas Pallant 28785959
- Xi Chen 31307124
- Nisha Devi 32726368

MONASH University

MD5, SHA-1

Weakness and Alternative

What is MD5 and SHA-1

MD5 (message-digest algorithm) is a cryptographic protocol used for authenticating messages as well as content verification and digital signatures. **MD5** is based on a hash function that verifies that a file you sent matches the file received by the person you sent it to. Previously, **MD5** was used for data encryption, but now it's used primarily for authentication. **MD5** was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities (Source Wiki).

SHA-1 or Secure Hash Algorithm 1 is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value. This hash value is known as a message digest. **SHA-1** is mainly used nowadays by SSL authorities to digital sign certificates in browser applications.

Weakness of MD5,SHA1

- **length extension attack**: The original hash methods are based upon hash functions using blocks of data. Hacker can take a hash for an unknown message, and then add additional data to produce a new valid hash.
- **Insufficient key length**: MD5 generates a message digest of 128-bits, while SHA1 generates a message digest of 160-bit hash value.
- MD5 and SHA-1 are many-to-one functions, which means multiple inputs can produce the same outputs. Thus, through the use of collision attack, it can reverse engineer the encryption function.

Alternatives

- SHA-256 is a more modern alternative encryption method to SHA-1, and used 256 bit encoding rather than SHA-1's 160 bits.
- Another alternative is the Chinese Hash function Whirlpool which uses 512 bit encryption, which would take hundreds of thousands of years of CPU compute time to break down.

SELF REFLECTION

The presentation could have been improved in many different ways. Firstly, some additional length about the history of SHA-1 and MD5 would have served well to understand why it came into popular use, and why it was preferred over other hash encryption methods.

Another area for improvement would be to go into detail about the methods of how MD5 and SHA-1 are hacked and made redundant, such as explaining how exactly a length

extension attack works and how an insufficient key length actually matters when it comes to decryption.

The last area for improvement I have identified is including more alternatives for SHA-1 and MD5, including explaining exactly how each of the alternatives are better than SHA-1 and MD5.

TASK 7.2

SCREENSHOTS

Startup.Auth.cs | dbo.AspNetUsers [Data] | StudentsController.cs | Create.cshtml | _Layout.cshtml

FIT5032_WebAuthentication | FIT5032_WebAuthentication.Startup | ConfigureAuth(IAppBuilder app)

```
using System;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.Google;
using Owin;
using FIT5032_WebAuthentication.Models;

namespace FIT5032_WebAuthentication
{
    2 references
    public partial class Startup
    {
        // For more information on configuring authentication, please visit https://go.microsoft.com/fwlink/?LinkId=301864
        1 reference
        public void ConfigureAuth(IAppBuilder app)
        {
            // Configure the db context, user manager and signin manager to use a single instance per request
            app.CreatePerOwinContext(ApplicationDbContext.Create);
            app.CreatePerOwinContext<ApplicationUserManager>(ApplicationUserManager.Create);
            app.CreatePerOwinContext<ApplicationSignInManager>(ApplicationSignInManager.Create);

            // Enable the application to use a cookie to store information for the signed in user
            // and to use a cookie to temporarily store information about a user logging in with a third party login provider
            // Configure the sign in cookie
            app.UseCookieAuthentication(new CookieAuthenticationOptions
            {
                AuthenticationType = DefaultAuthenticationTypes.ApplicationCookie,
                LoginPath = new PathString("/Account/Login"),
                Provider = new CookieAuthenticationProvider
                {
                    // Enables the application to validate the security stamp when the user logs in.
                    // This is a security feature which is used when you change a password or add an external login to your account.
                    OnValidateIdentity = SecurityStampValidator.OnValidateIdentity<ApplicationUserManager, ApplicationUser>(
                        validateInterval: TimeSpan.FromMinutes(30),
                        regenerateIdentity: (manager, user) => user.GenerateUserIdentityAsync(manager))
                }
            });
            app.UseExternalSignInCookie(DefaultAuthenticationTypes.ExternalCookie);

            // Enables the application to temporarily store user information when they are verifying the second factor in the two-factor authenti
```

62 % | No issues found | Ln: 1 | Ch: 1 | SPC | CRLF

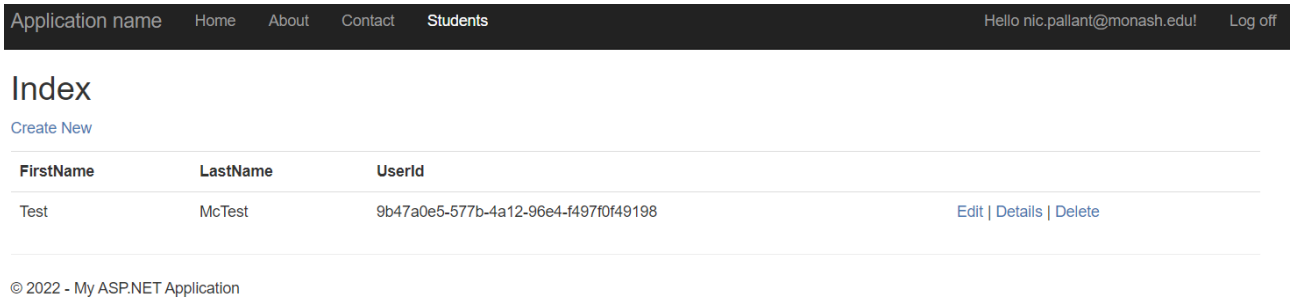
Application name | Home | About | Contact | Students | Register | Log in

Index

Create New

FirstName	LastName	UserId	
Test	McTest	9b47a0e5-577b-4a12-96e4-f497f0f49198	Edit | Details | Delete
Test2	LastNameTest	fe73af88-5499-422e-85ea-ce21618b1a83	Edit | Details | Delete

© 2022 - My ASP.NET Application

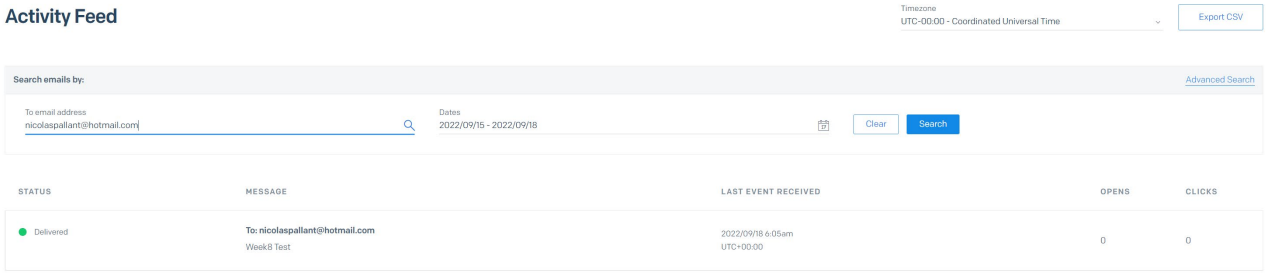


GIT REPOSITORY

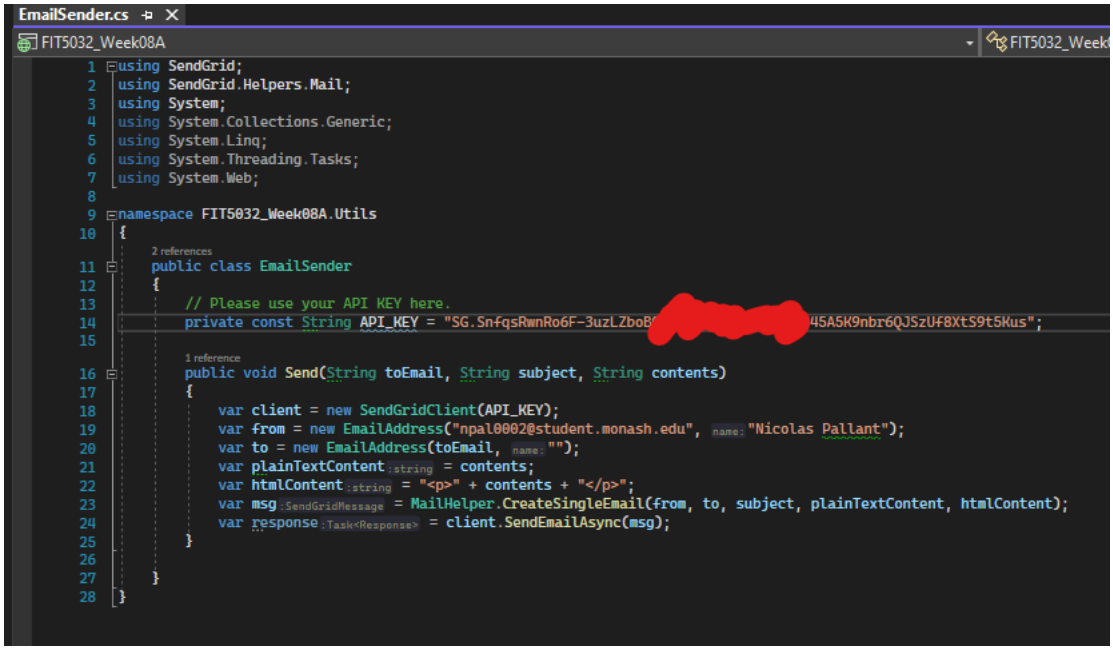
https://github.com/Ryukawastaken/FIT5032-Internet-Apps-Dev

TASK 8.1

SENDGRID EMAIL ACTIVITY SCREENSHOT



EMAILSENDER.CS SCREENSHOT WITH KEY



```
EmailSender.cs
FIT5032_Week08A
using SendGrid;
using SendGrid.Helpers.Mail;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;

namespace FIT5032_Week08A.Utils
{
    2 references
    public class EmailSender
    {
        // Please use your API KEY here.
        private const String API_KEY = "SG.SnfqsRwnRo6F-3uzLZboB[redacted]45A5K9nbr6QJSzUf8XtS9t5Kus";

        1 reference
        public void Send(String toEmail, String subject, String contents)
        {
            var client = new SendGridClient(API_KEY);
            var from = new EmailAddress("npal0002@student.monash.edu", name: "Nicolas Pallant");
            var to = new EmailAddress(toEmail, name: "");
            var plainTextContent :string = contents;
            var htmlContent :string = "<p>" + contents + "</p>";
            var msg :SendGridMessage = MailHelper.CreateSingleEmail(from, to, subject, plainTextContent, htmlContent);
            var response :Task<Response> = client.SendEmailAsync(msg);
        }

    }
}
```

GIT REPOSITORY

https://github.com/Ryukawastaken/FIT5032-Internet-Apps-Dev

TASK 8.2

ADVANTAGES OF SENDGRID

Using an external email company makes automation very convenient as an external company handles all of your automation for you, with you only having to set it up. You also don't have to host the web server that sends out the emails either.

On top of that, using an external company like Sendgrid allows for the convience of them collating all you metrics and performance data into one place that you can easily check and see how your emails are doing from month to month.

DISADVANTAGES OF SENDGRID

Having your emails automated through an external company like SendGrid opens your company up for security breaches. As your customer's email addresses will be forwarded to SendGrid, they will have access to them, regardless of whether or not they are encrypted.

Another disadvantage of SendGrid is that SendGrid will have access to pose as the email you have given them, which would allow them to spoof it and send emails on your half without your consent.