Research the email implementation and determine how it is supposed to work and why it currently does not

<u>How I'm going to approach this research task:</u>

1. Determine what the current structure for emails is in the codebase now

2. Diagnose why the current codebase does not send any emails

3. Determine how we can implement emails so that we can have active users join a hunt via an invitation link

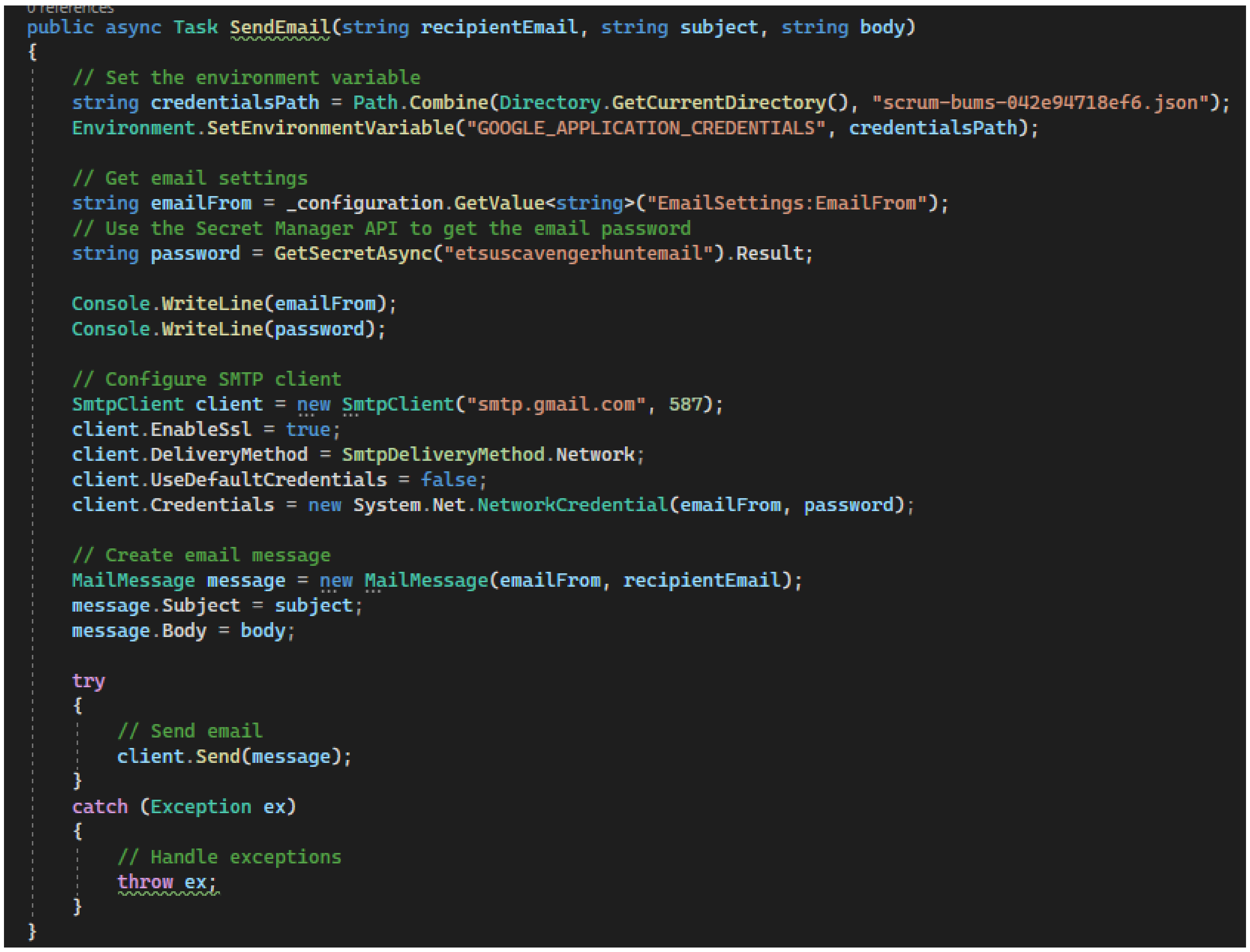
4. Document any problems we may run into in this process

5. Document any other solutions that may be easier if the current code has sub-optimal email implementation

6. I'm going to make an instance of the SendEmail code using a button that will just send an email to my account I have on the website and see what errors occur

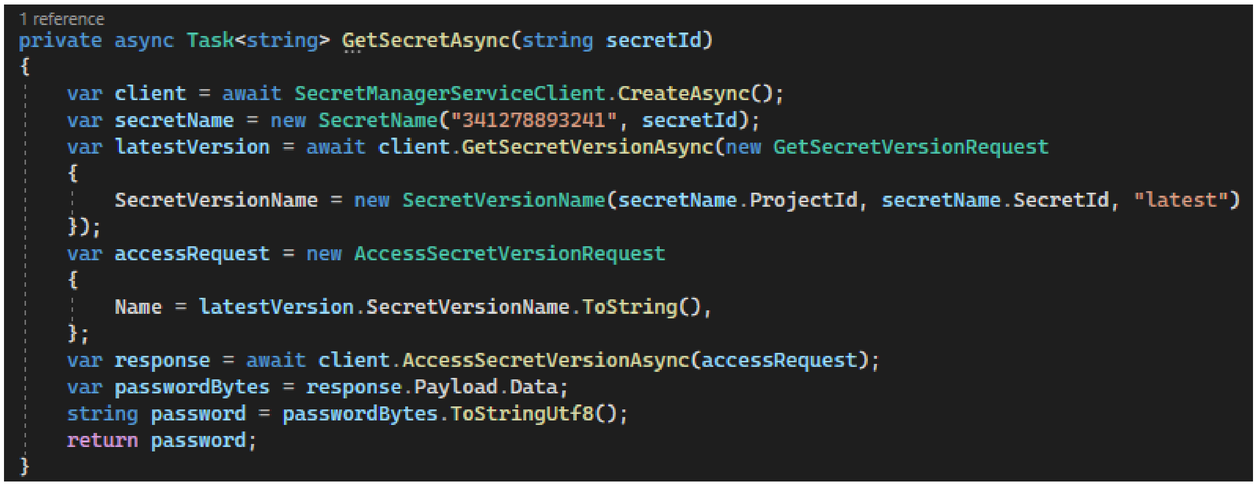
1. **Determine what the current implemented structure for emails exists in the codebase now**

So first I found where SendEmail was. (Services -> Functions.cs) and here is the code.



```
0 references
public async Task SendEmail(string recipientEmail, string subject, string body)
{
    // Set the environment variable
    string credentialsPath = Path.Combine(Directory.GetCurrentDirectory(), "scrum-bums-042e94718ef6.json");
    Environment.SetEnvironmentVariable("GOOGLE_APPLICATION_CREDENTIALS", credentialsPath);

    // Get email settings
    string emailFrom = _configuration.GetValue<string>("EmailSettings:EmailFrom");
    // Use the Secret Manager API to get the email password
    string password = GetSecretAsync("etsuscavengerhuntemail").Result;

    Console.WriteLine(emailFrom);
    Console.WriteLine(password);

    // Configure SMTP client
    SmtpClient client = new SmtpClient("smtp.gmail.com", 587);
    client.EnableSsl = true;
    client.DeliveryMethod = SmtpDeliveryMethod.Network;
    client.UseDefaultCredentials = false;
    client.Credentials = new System.Net.NetworkCredential(emailFrom, password);

    // Create email message
    MailMessage message = new MailMessage(emailFrom, recipientEmail);
    message.Subject = subject;
    message.Body = body;

    try
    {
        // Send email
        client.Send(message);
    }
    catch (Exception ex)
    {
        // Handle exceptions
        throw ex;
    }
}
```

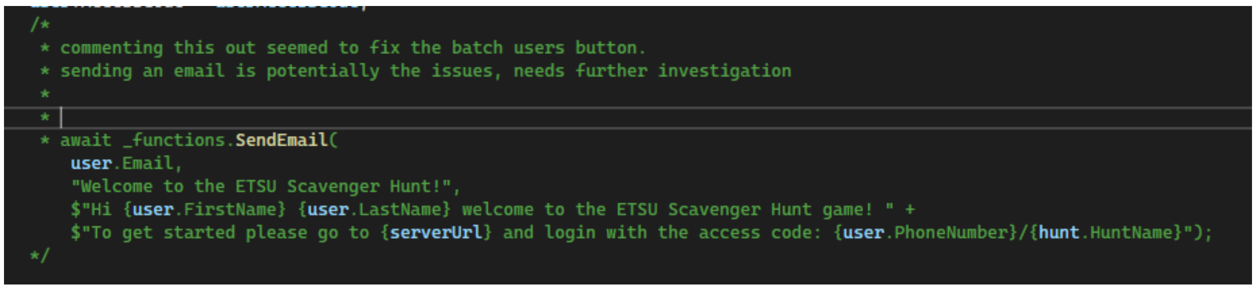
There was also this code that SendEmail is reliant upon (no comments)



```
1 reference
private async Task<string> GetSecretAsync(string secretId)
{
    var client = await SecretManagerServiceClient.CreateAsync();
    var secretName = new SecretName("341278893241", secretId);
    var latestVersion = await client.GetSecretVersionAsync(new GetSecretVersionRequest
    {
        SecretVersionName = new SecretVersionName(secretName.ProjectId, secretName.SecretId, "latest")
    });
    var accessRequest = new AccessSecretVersionRequest
    {
        Name = latestVersion.SecretVersionName.ToString(),
    };
    var response = await client.AccessSecretVersionAsync(accessRequest);
    var passwordBytes = response.Payload.Data;
    string password = passwordBytes.ToStringUtf8();
    return password;
}
```

//this is apparently creating a client and requesting a sync from an external source. Await.client is waiting for a response from the place it is calling out to.

Then I tried to find where SendEmail was implemented elsewhere in the code.



```
/*
 * commenting this out seemed to fix the batch users button.
 * sending an email is potentially the issues, needs further investigation
 *
 *
 * await _functions.SendEmail(
    user.Email,
    "Welcome to the ETSU Scavenger Hunt!",
    $"Hi {user.FirstName} {user.LastName} welcome to the ETSU Scavenger Hunt game! " +
    $"To get started please go to {serverUrl} and login with the access code: {user.PhoneNumber}/{hunt.HuntName}");
*/
```

This is the ONLY reference to SendEmail and this was what was causing the Batch Create User button to break the solution. I can only infer that whoever implemented this intended that when users were added to the active user list they would get an email stating “Welcome, join this hunt”

2. **Diagnose why the current codebase does not send any emails**

Well, based on 1. we can determine that emails aren't sent because SendEmail isn't implemented correctly let alone anywhere in the codebase as is.

3. **Determine how we can implement emails so that we can have active users join a hunt via an invitation link**

So long as the current SendEmail actually works, I think it would be best if we implemented this when a new hunt is created and then sends out an email to all active users. We can either set the code to send out an email 30 minutes prior to the start time (example) or we can send an email when the hunt is created and just state the start and end time in the email.

4. **Document any problems we may run into in this process**

So I don't think we'd run into many problems getting this setup, worst case we can just make our own email function since the current one doesn't seem to work and isn't referenced anywhere. If we do decide to stick with this solution we'd have to work around how the SecretManagerServicew works and then get the emails to send out how we want rather than trying to send them when accounts are added to the active user list.

5. **Document any other solutions that may be easier if the current code has sub-optimal email implementation**

So I found this video on youtube that explains how to send emails in C# / ASP.NET core
▶ Send Emails in C# and ASP.NET Core! - It´s actually pretty SIMPLE!

I watched the video and this is definitely along the lines of what they were implementing. They just didn't follow the steps at all, they got to like step 2 and stopped there.

How the email SHOULD communicate in the code;

Email Interface
Email Code as a function
Builder.Services.AddTransient<IEmail, Email>() **// every time we want to send an email a new instance of an email sender service is created. (Current solution has Scoped and Singleton which also work but they never did the <IEmail, Email> part and just added <Functions>**
Private email field in a Controller
An Index for who to send the emails out to (not bad to implement based on video we would just have to alter it for all active users)

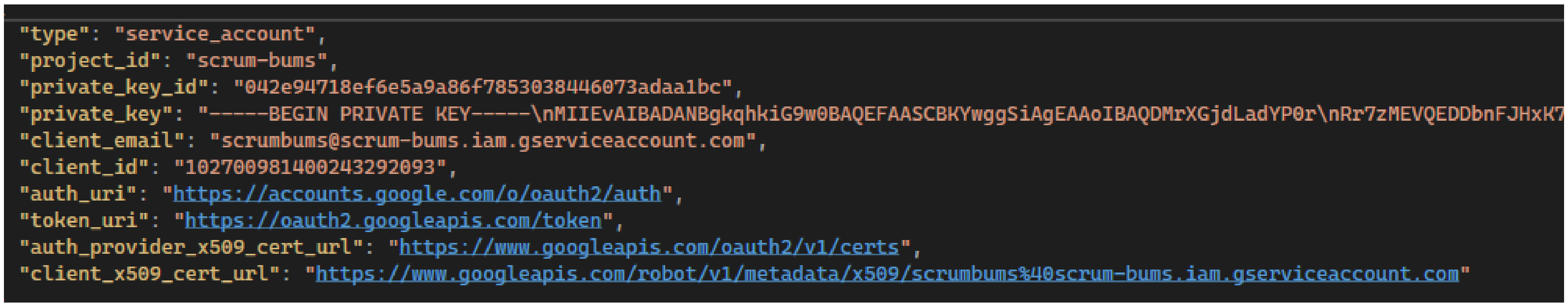
The only problem I ran into on the video was that when he ran his code the email was already sent out to the receiver so we would just have to find a way to send emails on command. This is where I think we should implement sending emails out when a hunt is created and/or when a hunt is about to reach its start time.

Summary:

I think what is in the code already is *okay.* It just feels like it needs to be hashed out a little more or completely redone since it's basically a waste of space at the moment. To fix the current implementation we would need to “update” or replace the current .json file that contains the login credentials.

6. **I’m going to make an instance of the SendEmail code using a button that will just send an email to my account I have on the website and see what errors occur**

The error that occurs has to do with credentials that the previous team implemented here



```
"type": "service_account",
"project_id": "scrum-bums",
"private_key_id": "042e94718ef6e5a9a86f7853038446073adaa1bc",
"private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDMrXGjdLadYP0r\nRr7zMEVQEDDbnFJHxK7
"client_email": "scrumbums@scrum-bums.iam.gserviceaccount.com",
"client_id": "102700981400243292093",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",
"token_uri": "https://oauth2.googleapis.com/token",
"auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
"client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/scrumbums%40scrum-bums.iam.gserviceaccount.com"
```