# **Current Status of Email Implementation**

## Question: Can C# send an email using a Gmail account

Yes, C# can be used to send emails through your Gmail account using the System.Net.Mail namespace. However, Gmail by default has security measures in place that might block access from less secure apps or require additional configurations to enable sending emails programmatically.

[Here's](#_Chat_GPT’s_Suggested) a basic example of how you can send an email using C# and Gmail's SMTP server.

* Remember to replace "yourgmail@gmail.com" with your Gmail address and "yourpassword" with your Gmail account password. Also, adjust the recipient's email address accordingly.
* Gmail might block sign-in attempts from apps it considers less secure. To allow C# to access your Gmail account, you may need to enable "Less Secure Apps" in your Google account settings. Alternatively, you can use OAuth authentication for a more secure method. <https://developers.google.com/identity/sign-in/web/sign-in>
* Gmail might require additional security configurations, like allowing access for "less secure apps" or setting up app-specific passwords. Check your Gmail settings for any necessary adjustments.
* For security reasons, avoid hardcoding sensitive information like passwords in your code. Consider using secure methods, such as environment variables or configuration files, to store and access sensitive data.
* Before using this code in a production environment, it's advisable to thoroughly test it and ensure that it complies with Google's security requirements and best practices.

## Chat GPT’s Suggested Template Code For Sending Emails from Gmail’s Server

using System;

using System.Net;

using System.Net.Mail;

class Program

{

    static void Main()

    {

        // Gmail credentials (replace with your Gmail address and password)

        string senderEmail = "sender@gmail.com";

        string senderPassword = "Password1";

        // Recipient's email address

        string recipientEmail = "recipient@gmail.com";

        // Create and configure the SMTP client

        SmtpClient smtpClient = new SmtpClient("smtp.gmail.com", 587);

        smtpClient.EnableSsl = true;

        smtpClient.UseDefaultCredentials = false;

        smtpClient.Credentials = new NetworkCredential(senderEmail, senderPassword);

        // Create the email message

        MailMessage mail = new MailMessage(senderEmail, recipientEmail);

        mail.Subject = "Test Email";

        mail.Body = "This is a test email sent from C#.";

        try

        {

            // Send the email

            smtpClient.Send(mail);

            Console.WriteLine("Test email sent successfully.");

        }

        catch (Exception ex)

        {

            Console.WriteLine("Failed to send test email. Error: " + ex.Message);

        }

    }

}

## Explanation of “Functions.cs.”

(Can be found under Services –> Functions.cs )

This code seems to be part of a system responsible for sending emails, specifically using Gmail's SMTP server, while keeping sensitive information like email passwords secure by using Google Secret Manager for storing and retrieving them. It demonstrates best practices for handling sensitive information and leveraging external services for secure credential storage.

The SMTP server for Gmail is a free SMTP server that anyone across the globe can use. It allows you to manage email transactions from your Gmail account via email clients or web applications. Email clients are user-end mail applications. Some of the most popular ones are Thunderbird, Outlook, and Mac Mail.

This code appears to be a C# class named **Functions**, which likely serves as a part of a larger application or service. Here's a breakdown of what the code does:

1. **Constructor with Configuration Injection:** The **Functions** class has a constructor that accepts an **IConfiguration** parameter injected through dependency injection. This allows it to access configuration settings from different sources, such as appsettings.json or environment variables.
2. **SendEmail Method:** This method (**SendEmail**) is responsible for sending an email using Gmail's SMTP server. It takes the recipient's email address, email subject, and email body as parameters.
   * It sets the environment variable **GOOGLE\_APPLICATION\_CREDENTIALS** to point to a JSON file (**scrum-bums-042e94718ef6.json**) containing Google Cloud credentials. This file is likely used for authentication purposes when interacting with Google services.
   * Retrieves email settings (**EmailFrom**) and password from the configuration using **\_configuration.GetValue<string>("EmailSettings:EmailFrom")**. The email sender's address is obtained from the configuration, and the email password is retrieved using the Google Secret Manager API, likely to keep the password secure and not hardcoded in the code.
   * Configures an SMTP client (**SmtpClient**) to send emails through Gmail's SMTP server using SSL. It uses the retrieved email address and password for authentication.
   * Creates a **MailMessage** object with the provided recipient email, subject, and body.
   * Attempts to send the email using the configured SMTP client (**client.Send(message)**), catching any exceptions that might occur during the sending process.
3. **GetSecretAsync Method:** This method retrieves the email password from Google Secret Manager using the provided **secretId**. It makes an asynchronous call to Secret Manager, gets the latest secret version, and extracts the password data.
   * It initializes a **SecretManagerServiceClient** and retrieves the latest version of the specified secret (**etsuscavengerhuntemail**) from Secret Manager. The retrieved secret is likely the email password.
   * It decodes the retrieved password data from bytes to a string (**passwordBytes.ToStringUtf8()**) and returns it.

## Action Steps:

1. **Decide with team:** When do the emails need to be sent?
   1. How many email templates will we need?
      1. Upon joining a team/ being added to a team?
      2. Upon the hunt being created?
      3. Upon the hunt starting/ending?
2. Decide if we need to recreate functionality or use existing functionality.
   1. **Note:** The “Batch User Button” used some kind of email functionality. It appeared that emails would be sent when clicking on the button.
3. Regroom/reevaluate this task.