Task 1 Title: Development of Email Notification Feature POC

Objective:

To create a proof of concept (POC) for an email notification feature within an application, enabling it to send automated emails to users under specified conditions (e.g., registration confirmation, password reset).

Background:

Email notifications are a critical component of user engagement and communication strategy in web applications. This feature should demonstrate the capability to dynamically generate and send emails to users based on specific triggers within the application.

Email Delivery Service or SMTP server:

The Simple Mail Transfer Protocol (SMTP) is an application used by mail servers to send, receive, and relay outgoing email between senders and receivers.

As the technology behind email communication, SMTP is the protocol that allows us to send and receive emails. Without it, email communication would be nonexistent since SMTP determines which servers will receive your relay messages.

There are various free and paid email delivery services available in the market which provide different functionalities. I have used **MailKit** for this task.

Configuration Process for SMTP Server:

1. **SMTP Server Configuration:**
    - First of all, we need to obtain SMTP server details from email service provider. I have used gmail account credentials.
    - We need to add SMTP server settings including host, port, username, password in the **appsettings.json** file of the ASP.NET project for backend.

```
"EmailHost": "smtp.gmail.com",
"Sendername": "noreply",
"EmailUsername": "ashutoshrai387@gmail.com",
"EmailPassword": "qaloqecgphqvxkxp",
```

2. **SMTP Client Setup:**
   - I have used MailKit to interact with the SMTP server.
   - I configured the SMTP client with the settings from the configuration file.
   - I used controllers, models, services and interfaces in the backend api for the configuration.

```
using var smtp = new SmtpClient();
smtp.Connect(_config.GetSection("EmailHost").Value, 587, MailKit.Security.SecureSocketOptions.StartTls);
smtp.Authenticate(_config.GetSection("EmailUsername").Value, _config.GetSection("EmailPassword").Value);
smtp.Send(email);
smtp.Disconnect(true);
```

Creating and Modifying Email Templates:

- I created a separate folder for templates and stored my templates there.
- I have defined the template structures with placeholder for dynamic content.
- I am using HTML templates as they offer customization options.
- I used variables to inject dynamic content into email templates and replaced variables with actual data during email generation.

```
email.From.Add(senderAddress);
email.To.Add(MailboxAddress.Parse(receiverEmail));
email.Subject = "Password Reset Request";

//Using the templates for email body
string serviceDirectory = Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
string templatePath = Path.Combine(serviceDirectory, "..", "..", "..", "Templates", "resetPassword.html");
templatePath = Path.GetFullPath(templatePath);
if (File.Exists(templatePath))
{
    string htmlTemplate = File.ReadAllText(templatePath);
    string replacedTemplate = htmlTemplate.Replace("{toNamePlaceholder}", receiverName);

    email.Body = new TextPart(MimeKit.Text.TextFormat.Html)
    {
        Text = replacedTemplate
    };
}
else
{
    Console.WriteLine("The specified template file does not exist.");
}

using var smtp = new SmtpClient();
```

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title></title>
</head>
<body>
    <p>Dear {toNamePlaceholder},</p>
    <p>We are delighted to inform you that your registration was successful!</p>
    <p>With your registration complete, you now have full access to all the features and services we offer. We're here to ensure
    <p>We're thrilled to have you and look forward to seeing you thrive with us. If you have any questions, feedback, or concern
    <p>Best regards,<br>Team.</p>
</body>
</html>
```

Integrating Triggers for Email Notifications in Angular and API Backend:

- I defined the events for user registration and password reset requests in Angular frontend and API backend to trigger email notifications.
- Implemented the event handling mechanisms in Angular application to detect trigger events using Angular components & event emitters.
- The relevant data associated with each trigger event is captured and passed to the backend API for email notification processing.
- I have used API endpoints in the backend application to receive requests from the Angular frontend when trigger events occur.
- I used Angular's HttpClient module to send HTTP requests from the frontend to the backend API endpoints corresponding to trigger events.

```
export class SignupComponent {
  regData: any = {};
  constructor(private http:HttpClient) {}
  signUp() {
    this.http.post<any>('https://localhost:7214/api/EmailNotification', this.regData)
      .subscribe(
        response => {
          console.log('Registration successful:', response);
          window.alert('Registration Successful');
        },
        error => {
          console.error('Registration failed:', error);
          window.alert('Registration Failed');
        }
      );
  }
```

Testing the Application

I tested the integration between the Angular frontend and API backend by triggering the events for Registration & Password Reset and the email notifications were sent successfully. The email notifications supported dynamic content.