Development of Email Notification Feature POC

Feature Name :
Development of Email Notification Feature POC

1. Requirement Analysis
The feature aims to implement email notifications within the application, enabling automated emails to be sent to users based on specific action, such as registration confirmation and password reset.

2.Impact Analysis
Analysis: The feature introduces a new functionality of sending emails based on user authentication event. It impacts the user registration, verification, login, and password reset flows. It requires integration with an email delivery service or SMTP server and handling of email templates and delivery status.

Current Schema Overview:

- Id: An integer representing the unique identifier of the user.
- Email: A string representing the email address of the user.
- PasswordHash: A byte array representing the hashed password of the user.
- PasswordSalt: A byte array representing the salt used in password hashing.

Proposed Changes:

- Added VerifiedAt: A nullable DateTime field to track the verification status of user accounts.
- Added PasswordResetToken: A nullable string field to facilitate the password reset process.
- Added ResetTokenExpires: A nullable DateTime field indicating the validity period of the password reset token.
- Added PasswordVerificationToken: A nullable string field representing the token used for password verification.

4.API Endpoints Design

Endpoint Summary:

- POST /api/user/register: Registers a new user and sends a verification email.
- POST /api/user/verify: Verifies a user account based on a token.
- POST /api/user/login: Logs in a user and sends a login success email.
- POST /api/user/forgot-password: Initiates the password reset process and sends a password reset email.
- POST /api/user/reset-password: Resets a user's password and sends a password reset confirmation email.
- GET /api/user: Get all the users
- DELETE api/user/:id : Delete specific user

5. Pseudo Code for Key Functionalities

Function Overview

- User Registration: Handles the registration process for new users, including checking for existing users, creating password hashes, and sending verification emails.
- Functionality: User Registration
- Function Name: Register

Development of Email Notification Feature POC

Pseudo Code

```
FUNCTION RegisterUser(request)
    TRY
        IF UserExists(request.Email) THEN
            RETURN "User already exists"
        END IF

      GeneratePasswordHash(request.Password, passwordHash,
passwordSalt)

        CREATE user AS User

        SaveUserToDatabase(user)

        CREATE emailDto AS EmailDto

        SendVerificationEmail(emailDto)

        RETURN "Registration successful"
    CATCH
        RETURN "An error occurred during registration"
    END TRY
END FUNCTION
```

User Verification: Verifies user accounts using a provided token, updating their verification status in the database, and sending a success email upon verification.

- Functionality: User Verification
- Function Name: Verify

Pseudo Code

```
FUNCTION VerifyUser(token)
    TRY
        SET user = GetUserByToken(token)
```

Development of Email Notification Feature POC

```
        IF user IS NULL THEN
            RETURN "Invalid token"
        END IF

        SET user.VerifiedAt = CurrentDateTime()
        SaveUserToDatabase(user)

        CREATE emailDto AS EmailDto
        SET emailDto.To = user.Email
        SET emailDto.Subject = "Account verified"
        SET emailDto.Body = "Your account has been successfully
verified."

        SendVerificationSuccessEmail(emailDto)

        RETURN "Account verified"
    CATCH
        LOG_ERROR("Error occurred during user verification")
        RETURN "An error occurred during verification"
    END TRY
END FUNCTION
```

User Login: Manages user login, verifying credentials, checking account verification status, and sending a login success email.

- Functionality: User Login
- Function Name: Login

Pseudo code

```
FUNCTION UserLogin(request)
    TRY
        SET user = GetUserByEmail(request.Email)

        IF user IS NULL THEN
            RETURN "User not found"
```

```
        END IF

            IF !VerifyPassword(request.Password,
    user.PasswordHash, user.PasswordSalt) THEN
                RETURN "Incorrect password"
        END IF

        IF !user.Verified THEN
            RETURN "Account not verified"
        END IF

        CREATE emailDto AS EmailDto
        SET emailDto.To = user.Email
        SET emailDto.Subject = "Login success"
        SET emailDto.Body = "Your account has been successfully
logged in."

        SendLoginSuccessEmail(emailDto)

        RETURN user
    CATCH
        RETURN "An error occurred during login"
    END TRY
END FUNCTION
```

Forgot Password: Handles password reset requests, generating reset tokens, sending reset instructions via email, and logging successful email sending.

- Functionality: Forgot Password
- Function Name: ForgotPassword

Pseudo Code

```
FUNCTION ForgotPassword(email)
    TRY
        SET user = GetUserByEmail(email)

        IF user IS NULL THEN
```

```
            RETURN "User not found"
        END IF

        SET user.ResetToken = GenerateRandomToken()
        SET user.ResetTokenExpires = CurrentDateTime() + 24
hours
        SaveUserToDatabase(user)

        CREATE emailDto AS EmailDto
        SendPasswordResetEmail(emailDto)

        RETURN "Password reset instructions sent"
    CATCH
        LOG_ERROR("Error occurred during password reset")
        RETURN "An error occurred during password reset"
    END TRY
END FUNCTION
```

Reset Password: Manages password resets, validating reset tokens, updating passwords, sending confirmation emails, and logging errors.
Functionality: Reset Password
Function Name: ResetPassword

Pseudo Code

```
FUNCTION ResetPassword(request)
    TRY
        SET user = GetUserByResetToken(request.Token)

        IF user IS NULL OR user.ResetTokenExpires <
CurrentDateTime() THEN
            RETURN "Invalid or expired token"
        END IF

        SET user.PasswordHash =
GeneratePasswordHash(request.NewPassword)
```

Development of Email Notification Feature POC

```
        SET user.ResetToken = NULL
        SET user.ResetTokenExpires = NULL
        SaveUserToDatabase(user)

        CREATE emailDto AS EmailDto
        SET emailDto.To = user.Email
        SET emailDto.Subject = "Password reset successful"
        SET emailDto.Body = "Your password has been successfully
reset."

        SendPasswordResetSuccessEmail(emailDto)

        RETURN "Password reset successful"
    CATCH
        LOG_ERROR("Error occurred during password reset")
        RETURN "An error occurred during password reset"
    END TRY
END FUNCTION
```

Pseudo Code for Sending Email Function
Functionality: Send Email
Function Name: SendEmail
Parameters:

request: EmailDto (An object contains To, subject, and body)

Pseudo Code:

```
FUNCTION SendEmail(request)
    TRY
        SET email = MimeMessage()

        email.From = Sender

        email.To = To
```

```
        email.Subject = Subject
        SET textPart = TextFormat.Html
        textPart.Text = Body
        email.Body = textPart

        // Connect to the SMTP server
        SET client = NEW SmtpClient()
        client.Connect(Server, Port, SecureSocketOptions.StartTls)

        // Authenticate with SMTP credentials
        client.Authenticate(Sender,Password)

        // Send the email
        client.Send(email)

        // Disconnect from the SMTP server
        client.Disconnect(true)

        // Return success message
        RETURN "Email sent successfully"

    CATCH
        // Log error
        LOG_ERROR("An error occurred while sending email")

        // Return error message
        RETURN "An error occurred while sending email"
    END TRY
END FUNCTION
```