Development of Email Notification Feature

By

Nisha Makwana,

Software Engineer Intern, at

Promact Infotech Private Limited

PROMACT

Table of contents:

1. Introduction
2. Research on Email Delivery Services
3. Research on Mailkit
4. Selection and setup
5. Development process
6. Usage
7. Testing and Validation
8. Conclusion

1. Introduction:

The Email Notification Feature is an essential aspect of our project, developed using ASP.NET Web API for the backend and Angular for the frontend. This report provides an overview of the development process, including setup, configuration, and usage of the feature. The feature comprises two functionalities: registration form notification and password reset notification.

2. Research on Email Delivery Services:

Before implementing the email notification feature, extensive research was conducted on different email delivery services to determine the most suitable option for our project. The following services were considered:

- **SendGrid**: Known for its reliability, scalability, and comprehensive features. It offers APIs for sending transactional and marketing emails, along with robust analytics and reporting capabilities.

- **Amazon SES (Simple Email Service**): Provides a reliable, scalable, and cost-effective email sending service. It offers features such as email authentication, bounce handling, and reputation monitoring.

- **Mailgun:** Offers a powerful set of APIs for sending, receiving, and tracking emails. It provides features like email validation, analytics, and spam filtering.

- **SMTP (Simple Mail Transfer Protocol):** A traditional method for sending emails using an SMTP server. While it lacks advanced features compared to dedicated email delivery services, it provides a basic, customizable solution.

3. Research on MailKit:

→ MailKit was selected as the library for email delivery after comprehensive research on various options available. The following aspects were considered during the research:

- **Features**: MailKit offers a wide range of features necessary for robust email delivery, including support for SMTP, IMAP, and POP3 protocols, MIME parsing and generation, SSL/TLS encryption, and message signing and encryption.

- **Community Support**: MailKit benefits from a strong and active open-source community, providing regular updates, bug fixes, and community-driven support forums.

- **Performance**: MailKit is known for its high performance and efficiency, making it suitable for applications with high email throughput requirements.

- **Documentation**: MailKit provides comprehensive documentation, including usage guides, API references, and code examples, facilitating easy integration and troubleshooting.

- **Compatibility**: MailKit is compatible with various email servers and platforms, ensuring seamless integration with different environments.

- **Security**: MailKit implements security best practices, including support for secure authentication methods and encryption protocols, ensuring the confidentiality and integrity of email communication.

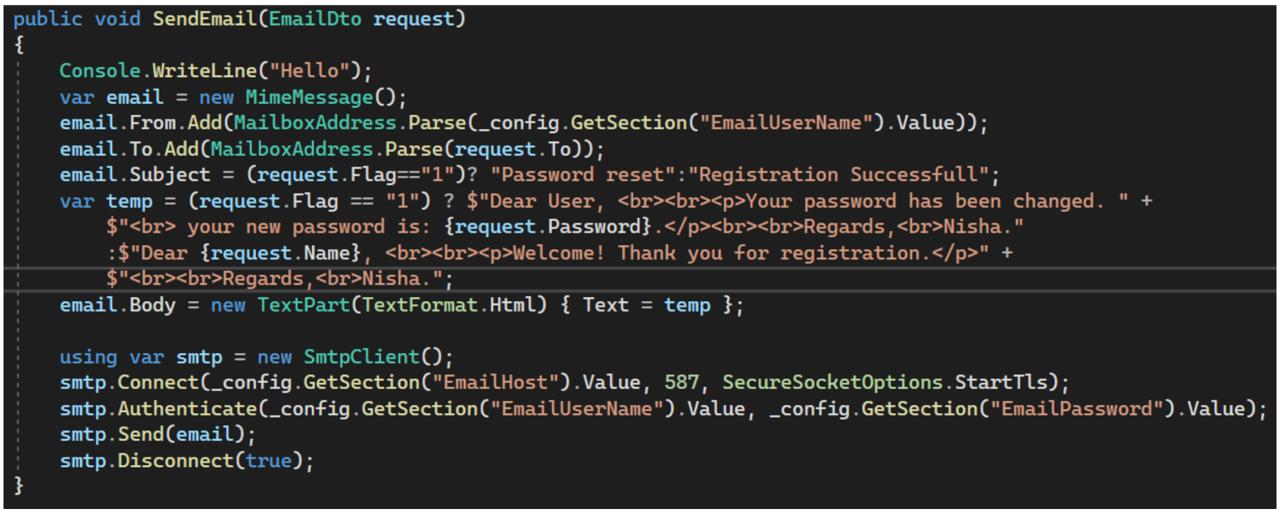
4. Selection and Setup:

After evaluating the different email delivery services, MailKit was chosen for its flexibility and ease of integration. The setup process involved:

- Mail Server Setup: Configured SMTP settings in the ASP.NET Web API project to connect to the mail server.

- Integration: MailKit was integrated into the project using NuGet Package Manager, ensuring that the latest version of the library was used.

- SMTP Configuration: SMTP settings were configured in the ASP.NET Web API project to connect to the mail server. This included specifying the SMTP host, port, credentials (username and password), and enabling SSL/TLS encryption if required.



```
public void SendEmail(EmailDto request)
{
    Console.WriteLine("Hello");
    var email = new MimeMessage();
    email.From.Add(MailboxAddress.Parse(_config.GetSection("EmailUserName").Value));
    email.To.Add(MailboxAddress.Parse(request.To));
    email.Subject = (request.Flag=="1")? "Password reset":"Registration Successfull";
    var temp = (request.Flag == "1") ? $"Dear User, <br><br><p>Your password has been changed. " +
        $"<br> your new password is: {request.Password}.</p><br><br>Regards,<br>Nisha."
        :$"Dear {request.Name}, <br><br><p>Welcome! Thank you for registration.</p>" +
        $"<br><br>Regards,<br>Nisha.";
    email.Body = new TextPart(TextFormat.Html) { Text = temp };

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

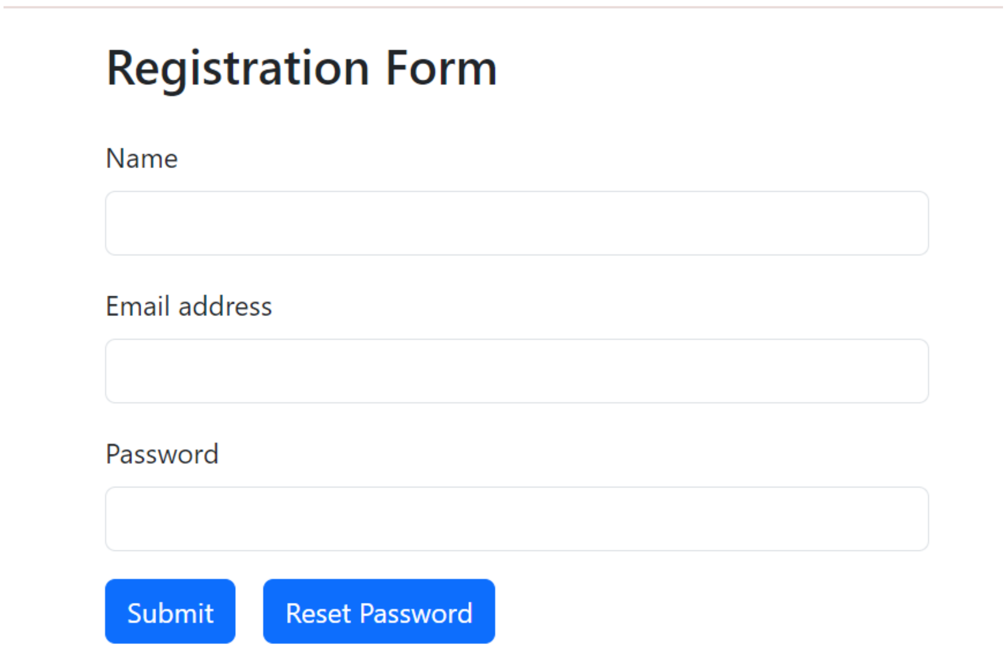
5. Development Process:

→Backend Development (ASP.NET Web API):

- Created logic to handle registration and password reset requests.
- Integrated email sending functionality using MailKit library.
- Implemented logic to trigger emails upon successful registration and password reset.

→Frontend Development (Angular):

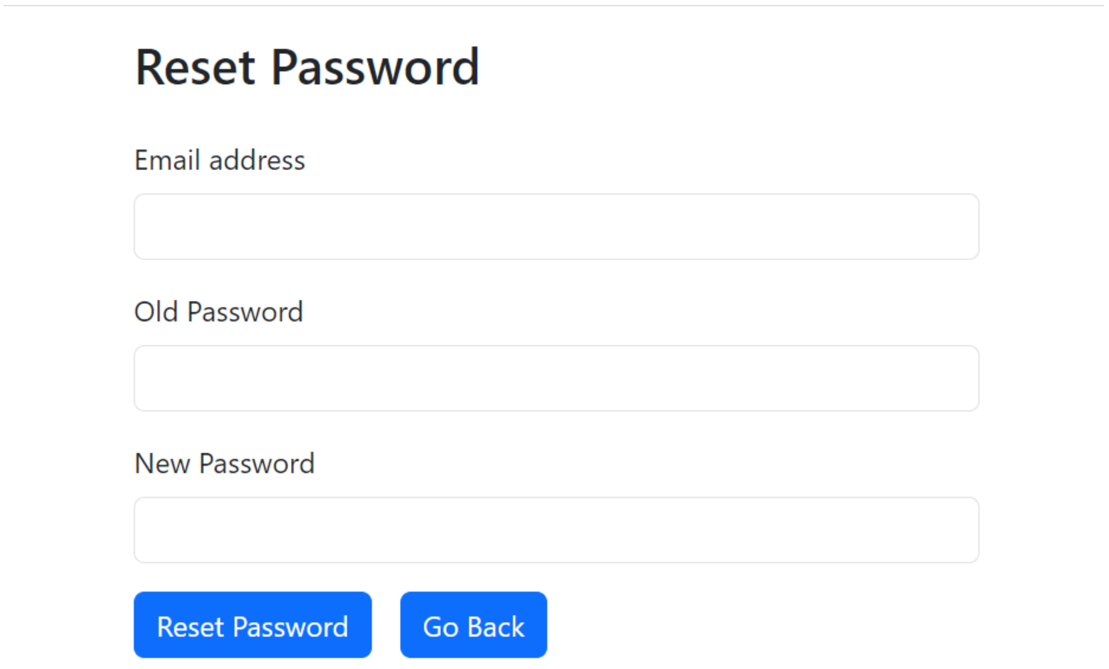
- Designed registration and password reset forms using Angular template-driven Forms.
- Implemented form submission logic to send data to the backend API endpoints.
- Provided user feedback upon successful submission and error handling for failed requests.


Registration Form

Name

Email address

Password



6. Usage:

Registration Form Notification:

- User fills out the registration form and submits it.
- Backend API receives the registration data, processes it, and triggers an email notification to the registered user.
- Users receive an email confirming their successful registration with relevant details.

Password Reset Notification:

- User fills out the password reset form with their email, old password, and new password.
- Backend API validates the request, updates the password, and triggers an email notification to the user.
- Users receive an email informing them that their password has been successfully reset.

7. Testing and Validation:

Unit Testing:

- Conducted unit tests for backend API endpoints to ensure proper functionality and email triggering.

Integration Testing:

- Integration testing was performed to verify seamless communication between the frontend and backend.

End-to-End Testing:

- The complete registration and password reset workflows were validated to ensure email notifications were triggered correctly.

8. Conclusion:

The Email Notification Feature, powered by MailKit, enhances user experience by providing timely notifications for registration and password reset actions. Through thorough research, setup, and development, the feature ensures reliable communication with users, contributing to the overall functionality and usability of the application.