The error System.IO.FileNotFoundException occurs because the file is not found at the specified path. This can happen if the file is not generated or saved correctly. To resolve this issue, you have **two main options**:

1. **Save the File to a Specific Folder**:
   * Save the file to a folder where the application has access.
   * If a file with the same name already exists, delete it before saving the new file.
2. **Generate the File in Memory and Attach It Directly**:
   * Generate the Excel file in memory (using a MemoryStream) and attach it to the email without saving it to disk.

Below, I'll provide the implementation for both approaches.

**Option 1: Save the File to a Specific Folder**

**Steps:**

1. **Define a Folder Path**:
   * Use a folder within your application (e.g., wwwroot/attachments) to save the file.
2. **Delete Existing File**:
   * Check if a file with the same name exists and delete it before saving the new file.
3. **Save the File**:
   * Save the Excel file to the specified folder.
4. **Attach the File to the Email**:
   * Read the file from the folder and attach it to the email.

**Updated SendEmail Action**

csharp

Copy

[HttpPost]

public async Task<IActionResult> SendEmail(string to, string subject, string body, string rebuiltPartNum)

{

try

{

// Define the folder path for saving the file

var folderPath = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "attachments");

if (!Directory.Exists(folderPath))

{

Directory.CreateDirectory(folderPath); // Create the folder if it doesn't exist

}

// Define the file path

var filePath = Path.Combine(folderPath, $"{rebuiltPartNum}\_RebuiltParts\_Estimate.xlsx");

// Delete the file if it already exists

if (System.IO.File.Exists(filePath))

{

System.IO.File.Delete(filePath);

}

// Generate the Excel file

var memoryStream = await GenerateExcelFile(rebuiltPartNum);

await System.IO.File.WriteAllBytesAsync(filePath, memoryStream.ToArray());

// Send the email with the attached Excel file

var message = new MimeMessage();

message.From.Add(new MailboxAddress("Your Name", "your-email@example.com"));

message.To.Add(new MailboxAddress("", to));

message.Subject = subject;

message.Body = new TextPart("plain") { Text = body };

var attachment = new MimePart("application", "vnd.openxmlformats-officedocument.spreadsheetml.sheet")

{

Content = new MimeContent(System.IO.File.OpenRead(filePath)),

ContentDisposition = new ContentDisposition(ContentDisposition.Attachment),

ContentTransferEncoding = ContentEncoding.Base64,

FileName = Path.GetFileName(filePath)

};

var multipart = new Multipart("mixed");

multipart.Add(message.Body);

multipart.Add(attachment);

message.Body = multipart;

using (var client = new SmtpClient())

{

await client.ConnectAsync("smtp.example.com", 587, SecureSocketOptions.StartTls);

await client.AuthenticateAsync("your-email@example.com", "your-email-password");

await client.SendAsync(message);

await client.DisconnectAsync(true);

}

return Json(new { success = true });

}

catch (Exception ex)

{

// Log the exception (you can use a logging framework like Serilog or NLog)

Console.WriteLine($"Error sending email: {ex.Message}");

return Json(new { success = false, message = ex.Message });

}

}

private async Task<MemoryStream> GenerateExcelFile(string rebuiltPartNum)

{

var memoryStream = new MemoryStream();

using (var package = new ExcelPackage(memoryStream))

{

// Add your logic to generate the Excel file

var worksheet = package.Workbook.Worksheets.Add("Sheet1");

worksheet.Cells["A1"].Value = "Rebuilt Part Number";

worksheet.Cells["B1"].Value = rebuiltPartNum;

// Save the Excel package to the memory stream

package.Save();

}

memoryStream.Position = 0; // Reset the stream position to the beginning

return memoryStream;

}

**Option 2: Generate the File in Memory and Attach It Directly**

**Steps:**

1. **Generate the Excel File in Memory**:
   * Use a MemoryStream to generate the Excel file.
2. **Attach the File Directly**:
   * Attach the file from the MemoryStream without saving it to disk.

**Updated SendEmail Action**

csharp

Copy

[HttpPost]

public async Task<IActionResult> SendEmail(string to, string subject, string body, string rebuiltPartNum)

{

try

{

// Generate the Excel file in memory

var memoryStream = await GenerateExcelFile(rebuiltPartNum);

// Send the email with the attached Excel file

var message = new MimeMessage();

message.From.Add(new MailboxAddress("Your Name", "your-email@example.com"));

message.To.Add(new MailboxAddress("", to));

message.Subject = subject;

message.Body = new TextPart("plain") { Text = body };

var attachment = new MimePart("application", "vnd.openxmlformats-officedocument.spreadsheetml.sheet")

{

Content = new MimeContent(memoryStream),

ContentDisposition = new ContentDisposition(ContentDisposition.Attachment),

ContentTransferEncoding = ContentEncoding.Base64,

FileName = $"{rebuiltPartNum}\_RebuiltParts\_Estimate.xlsx"

};

var multipart = new Multipart("mixed");

multipart.Add(message.Body);

multipart.Add(attachment);

message.Body = multipart;

using (var client = new SmtpClient())

{

await client.ConnectAsync("smtp.example.com", 587, SecureSocketOptions.StartTls);

await client.AuthenticateAsync("your-email@example.com", "your-email-password");

await client.SendAsync(message);

await client.DisconnectAsync(true);

}

return Json(new { success = true });

}

catch (Exception ex)

{

// Log the exception (you can use a logging framework like Serilog or NLog)

Console.WriteLine($"Error sending email: {ex.Message}");

return Json(new { success = false, message = ex.Message });

}

}

private async Task<MemoryStream> GenerateExcelFile(string rebuiltPartNum)

{

var memoryStream = new MemoryStream();

using (var package = new ExcelPackage(memoryStream))

{

// Add your logic to generate the Excel file

var worksheet = package.Workbook.Worksheets.Add("Sheet1");

worksheet.Cells["A1"].Value = "Rebuilt Part Number";

worksheet.Cells["B1"].Value = rebuiltPartNum;

// Save the Excel package to the memory stream

package.Save();

}

memoryStream.Position = 0; // Reset the stream position to the beginning

return memoryStream;

}

**Key Differences Between the Two Approaches**

| **Feature** | **Save to Folder** | **Generate in Memory** |
| --- | --- | --- |
| **File Storage** | Saves the file to disk. | Does not save the file to disk. |
| **File Access** | File can be accessed later. | File exists only during the request. |
| **Performance** | Slightly slower due to disk I/O. | Faster as it avoids disk I/O. |
| **Use Case** | Useful if the file needs to be reused. | Useful for one-time file generation. |

**Recommendation**

* Use **Option 2 (Generate in Memory)** if the file is only needed for the email attachment and does not need to be saved permanently.
* Use **Option 1 (Save to Folder)** if the file needs to be accessed later or reused.

**Full Files**

Here are the full files for **Option 2 (Generate in Memory)**:

**RebuiltPartsController.cs**

csharp

Copy

using MailKit.Net.Smtp;

using MimeKit;

using MailKit.Security;

using OfficeOpenXml;

using System.IO;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

public class RebuiltPartsController : Controller

{

[HttpPost]

public async Task<IActionResult> SendEmail(string to, string subject, string body, string rebuiltPartNum)

{

try

{

// Generate the Excel file in memory

var memoryStream = await GenerateExcelFile(rebuiltPartNum);

// Send the email with the attached Excel file

var message = new MimeMessage();

message.From.Add(new MailboxAddress("Your Name", "your-email@example.com"));

message.To.Add(new MailboxAddress("", to));

message.Subject = subject;

message.Body = new TextPart("plain") { Text = body };

var attachment = new MimePart("application", "vnd.openxmlformats-officedocument.spreadsheetml.sheet")

{

Content = new MimeContent(memoryStream),

ContentDisposition = new ContentDisposition(ContentDisposition.Attachment),

ContentTransferEncoding = ContentEncoding.Base64,

FileName = $"{rebuiltPartNum}\_RebuiltParts\_Estimate.xlsx"

};

var multipart = new Multipart("mixed");

multipart.Add(message.Body);

multipart.Add(attachment);

message.Body = multipart;

using (var client = new SmtpClient())

{

await client.ConnectAsync("smtp.example.com", 587, SecureSocketOptions.StartTls);

await client.AuthenticateAsync("your-email@example.com", "your-email-password");

await client.SendAsync(message);

await client.DisconnectAsync(true);

}

return Json(new { success = true });

}

catch (Exception ex)

{

// Log the exception (you can use a logging framework like Serilog or NLog)

Console.WriteLine($"Error sending email: {ex.Message}");

return Json(new { success = false, message = ex.Message });

}

}

private async Task<MemoryStream> GenerateExcelFile(string rebuiltPartNum)

{

var memoryStream = new MemoryStream();

using (var package = new ExcelPackage(memoryStream))

{

// Add your logic to generate the Excel file

var worksheet = package.Workbook.Worksheets.Add("Sheet1");

worksheet.Cells["A1"].Value = "Rebuilt Part Number";

worksheet.Cells["B1"].Value = rebuiltPartNum;

// Save the Excel package to the memory stream

package.Save();

}

memoryStream.Position = 0; // Reset the stream position to the beginning

return memoryStream;

}

}