Development of Import Page Content Feature POC:

1. Introduction

The "Impsort Page Content as PDF" feature was developed to allow users to import the content of a web page and export it as a PDF document. This report outlines the process of developing the feature, including the research, setup, development, testing, and documentation.

2. Feature Overview

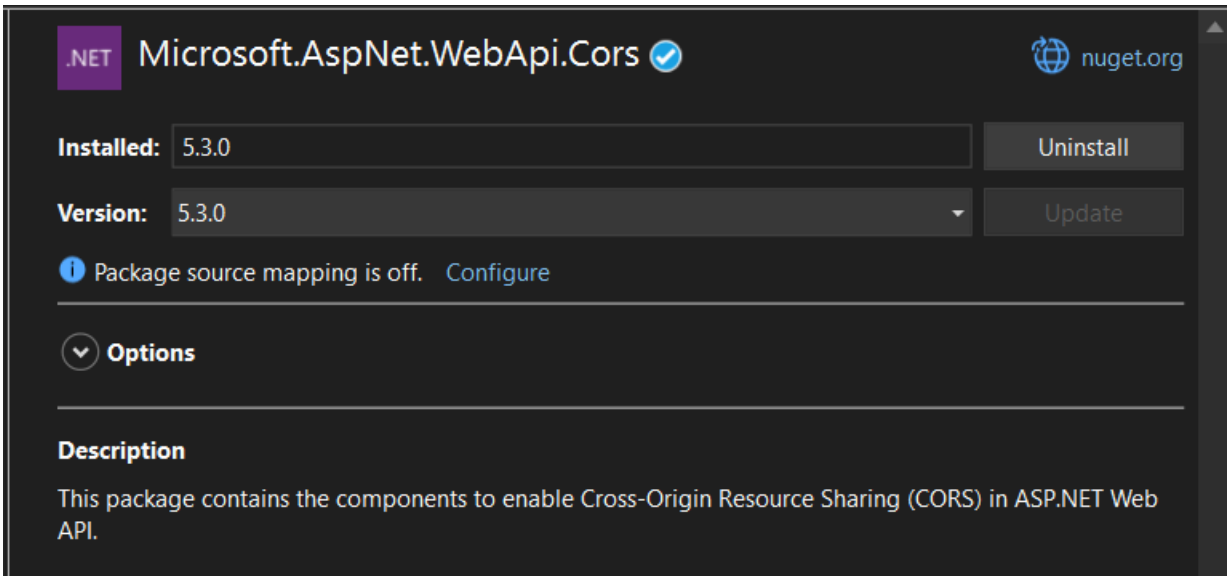
The "Import Page Content as PDF" feature will provide users with the ability to convert web pages or other content into a PDF document. This feature will be accessible through the application's user interface and will involve the following steps:

- **User Input:** Users will provide the URL of the web page or upload a file containing the content they wish to convert to PDF.
- **Conversion Process:** The application will process the input and convert the content into a PDF document.
- **Download or Share:** Once the conversion is complete, users will have the option to download the PDF document or share it via email or other means.

3. Research and setup

The research phase involved evaluating different HTML to PDF conversion libraries or APIs to find the most suitable option for the project's requirements. PuppeteerSharp was selected as the library for converting HTML content to PDF due to its robust features and ease of use. Additionally, CORS (Cross-Origin Resource Sharing) was configured to avoid CORS errors, which are common when using headless browsers like PuppeteerSharp to access resources from different domains. This setup ensures that PuppeteerSharp can access the necessary resources without encountering any CORS issues.

```
//Adding cors configration
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAnyOrigin",
          builder =>
          {
              builder.AllowAnyOrigin()
                     .AllowAnyMethod()
                     .AllowAnyHeader();
          });
});
```



PuppeteerSharp Configration

```
//Get Method is used for receiving getPDF and returning pdf
[HttpGet]
0 references
public async Task<IActionResult> GetPdf(string url)
{
    try
    {
        var browserFetcher = new BrowserFetcher();
        await browserFetcher.DownloadAsync();

        await using var browser = await Puppeteer.LaunchAsync(new LaunchOptions { Headless = true });

        await using var page = await browser.NewPageAsync();
        await page.GoToAsync(url, new NavigationOptions { Timeout = 30000, WaitUntil = new[] { WaitUntilNavigation.Networkidle0 } });

        var pdfStream = await page.PdfStreamAsync();

        _logger.LogInformation("PDF generated successfully.");

        return new FileStreamResult(pdfStream, "application/pdf");
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "An error occurred while generating the PDF.");
        return StatusCode(500, "An error occurred while generating the PDF.");
    }
}
```

4. Technical Implementation

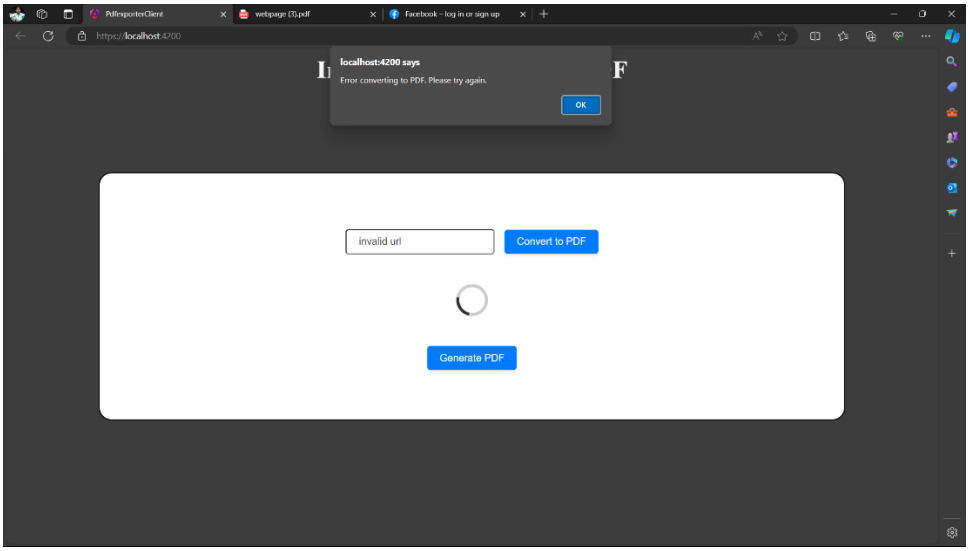
The technical implementation of the "Import Page Content as PDF" feature will involve the following components:

- **Frontend Interface:** The application's frontend, built using Angular, will include a user interface for users to input the URL or upload a file for conversion.
- **Backend Service:** The backend service, built using .NET, will handle the conversion process, using PuppeteerSharp to convert the content into a PDF document.
- **PDF Generation Library:** PuppeteerSharp will be used to convert the content into a PDF format. This library will handle the conversion process and provide the resulting PDF document.
- **Download or Share Options:** Once the PDF document is generated, the backend service will provide options for users to download the document or share it via email or other means.

5. Considerations

When implementing the "Import Page Content as PDF" feature, the following considerations should be taken into account:

- **Security:** Ensure that the conversion process is secure and does not expose the application to security vulnerabilities.
- **Performance:** Optimize the conversion process for performance, especially when handling large or complex content. (max 30 sec wait to convert into pdf otherwise throw error)
- **User Experience:** Design the user interface to be intuitive and user-friendly, guiding users through the conversion process.
- **Error Handling:** Implement error handling to gracefully handle any issues that may arise during the conversion process.
- **Cost:** Consider the cost implications of using a PDF generation library or service, especially if the application will be generating a large number of PDF documents.



6. Integration and Testing:

The integration phase involved integrating the HTML to PDF conversion functionality with the frontend interface. The feature was tested with various web pages to ensure the PDF output accurately reflected the web page content, including complex layouts and images.