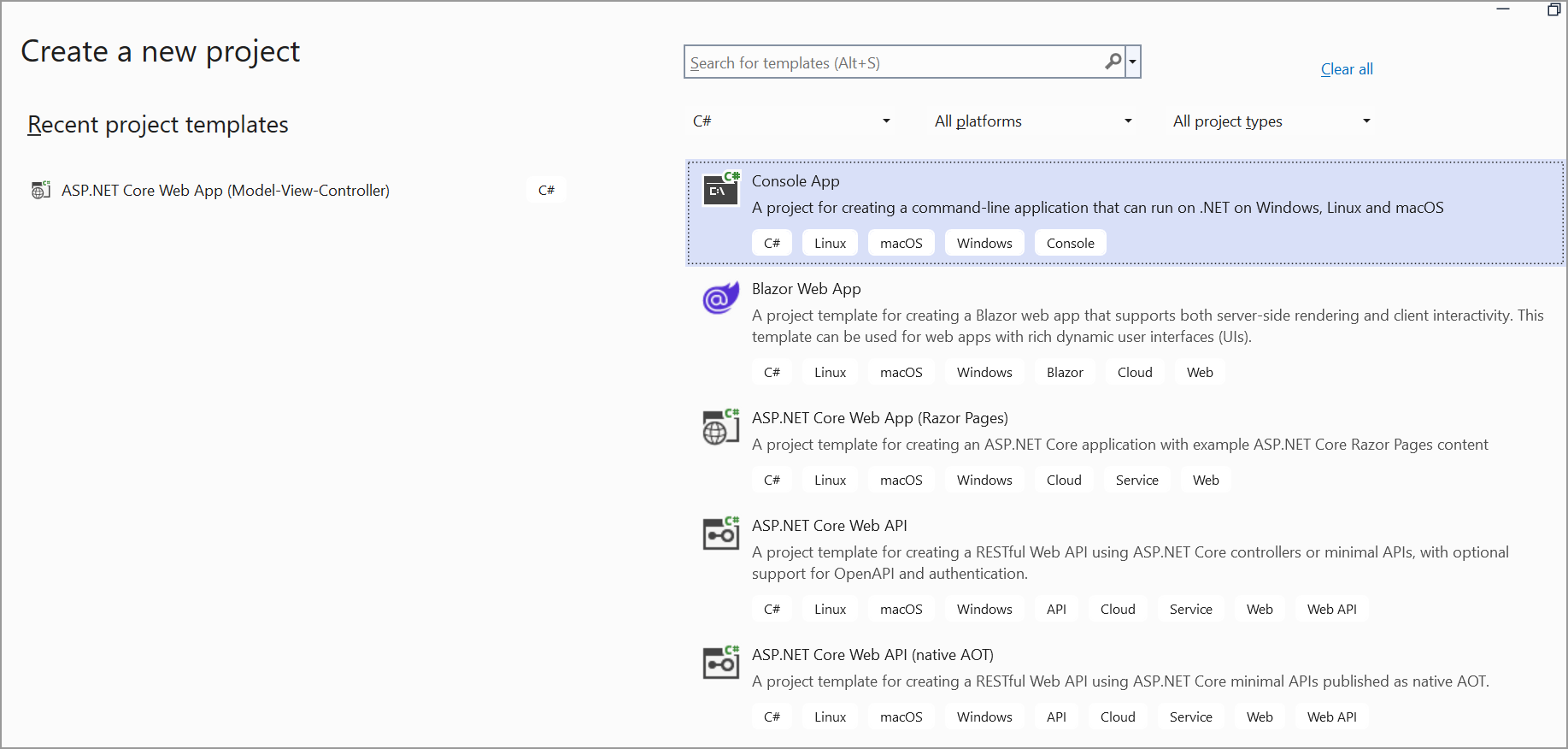
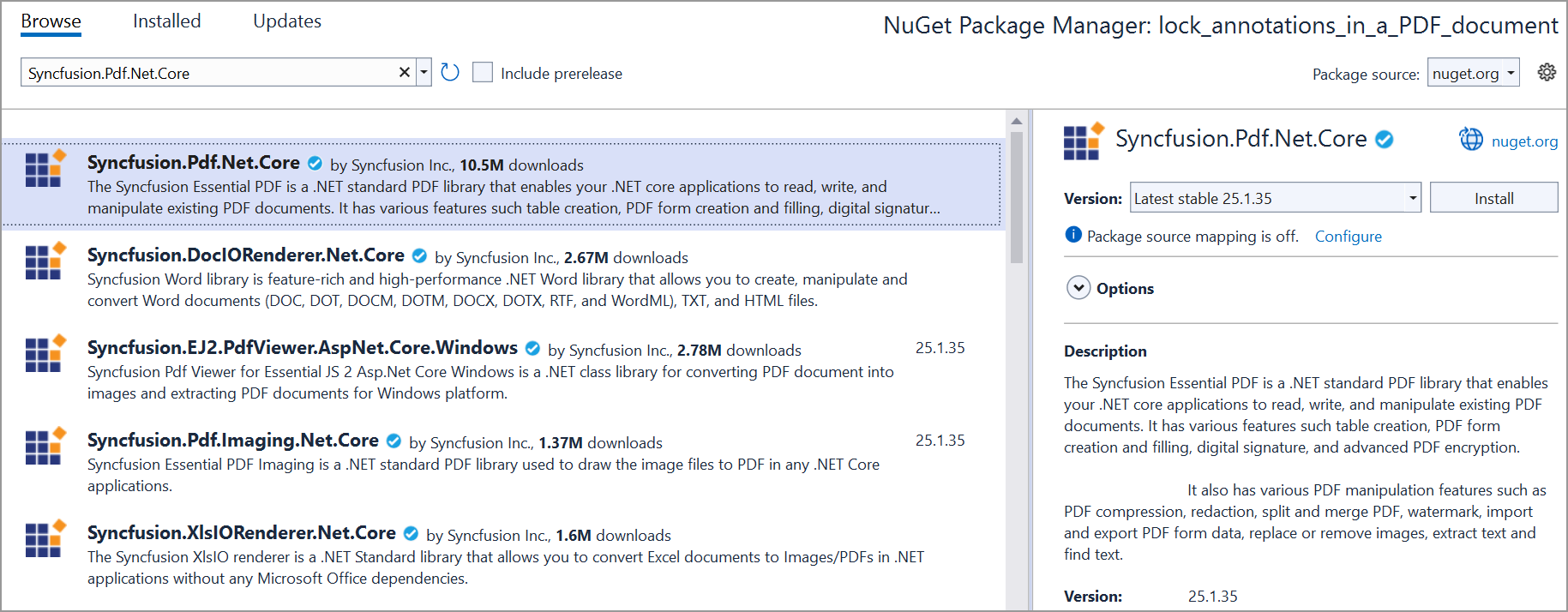
**How to Create the Automatic Bookmarking in PDF**

The Syncfusion Essential® PDF is a feature-rich and high performance [**.NET PDF library**](https://www.syncfusion.com/document-processing/pdf-framework/net) used to create, read, and edit PDF documents programmatically without Adobe dependencies. Using this library, you can create the automatic Bookmarking in PDF document using C#.

In this article, we have created a bookmark based on the font size which is used in the PDF document by extracting its text and font size.

**Steps to create the automatic Bookmarking in PDF document programmatically:**

1. Create a new console application project.
2. Install the **[Syncfusion.Pdf.Net.Core](https://www.nuget.org/packages/Syncfusion.Pdf.Net.Core" \t "_blank)** NuGet package as a reference to your console application from [**Nuget.org**](https://www.nuget.org/).



1. Include the following namespaces in the Program.cs file.

**C#**

using **Syncfusion**.Drawing;

using **Syncfusion**.Pdf;

using **Syncfusion**.Pdf.Interactive;

using **Syncfusion**.Pdf.Parsing;

1. Use the following code sample in Program.cs to create an automatic bookmark.

**C#**

// Open the PDF file using a file stream.

**FileStream** pdfFilePath = **new** **FileStream**(@"..\..\..\Data\Sample.pdf", **FileMode**.Open, **FileAccess**.Read);

// Using statement ensures proper disposal of resources after use.

using (**PdfLoadedDocument** loadedDocument = **new** **PdfLoadedDocument**(pdfFilePath))

{

// Iterate through each page in the loaded PDF document.

**for** (int i = 0; i < loadedDocument.Pages.Count; i++)

{

// Get the current page.

**PdfPageBase** page = loadedDocument.Pages[i];

// Initialize a collection to store text lines extracted from the page.

**var** lineCollection = **new** **TextLineCollection**();

// Extract text from the page and get a collection of text lines.

string extractedText = page.**ExtractText**(out lineCollection);

// Initialize a variable to store the current bookmark.

**PdfBookmark** bookmark = null;

//Iterate the text line collection to create bookmarks.

foreach (**TextLine** line **in** lineCollection.TextLine)

{

**if** (line.Text != null && line.Text != " ")

{

float fontSize = **GetFontSize**(line);

**if** (fontSize > 15)

{

// Create a bookmark for header 1.

string header1 = line.Text;

bookmark = loadedDocument.Bookmarks.**Add**(header1);

bookmark.Destination = **new** **PdfDestination**(page);

bookmark.Destination.Location = **new** **PointF**(line.Bounds.X, line.Bounds.Y);

bookmark.TextStyle = **PdfTextStyle**.Bold;

}

**else** **if** (fontSize >= 12 && fontSize <= 15 && bookmark != null)

{

// Create a bookmark for header 2.

string header2 = line.Text;

**PdfBookmark** childBookmark = bookmark.**Add**(header2);

childBookmark.Destination = **new** **PdfDestination**(page);

childBookmark.Destination.Location = **new** **PointF**(line.Bounds.X, line.Bounds.Y);

childBookmark.TextStyle = **PdfTextStyle**.Bold;

}

}

}

}

// Create a memory stream to save the modified PDF document.

**MemoryStream** ms = **new** **MemoryStream**();

// Save the modified PDF document to the memory stream.

loadedDocument.**Save**(ms);

// Write the contents of the memory stream to a file named "output.pdf".

**File**.**WriteAllBytes**("output.pdf", ms.**ToArray**());

// Close the loaded document.

loadedDocument.**Close**(true);

}

**static** float **GetFontSize**(**TextLine** textLine)

{

// Get a collection of words in the current line.

**List**<**TextWord**> textWordCollection = textLine.WordCollection;

float fontSize = 0;

//Iterate through each word in the current line.

foreach (**var** word **in** textWordCollection)

{

//Iterate through each glyph in the word.

foreach (**var** glyph **in** word.Glyphs)

{

//Get the font size of the glyph.

**if** (glyph.FontSize > 0)

{

fontSize = glyph.FontSize;

**break**;

}

}

**if** (fontSize != 0)

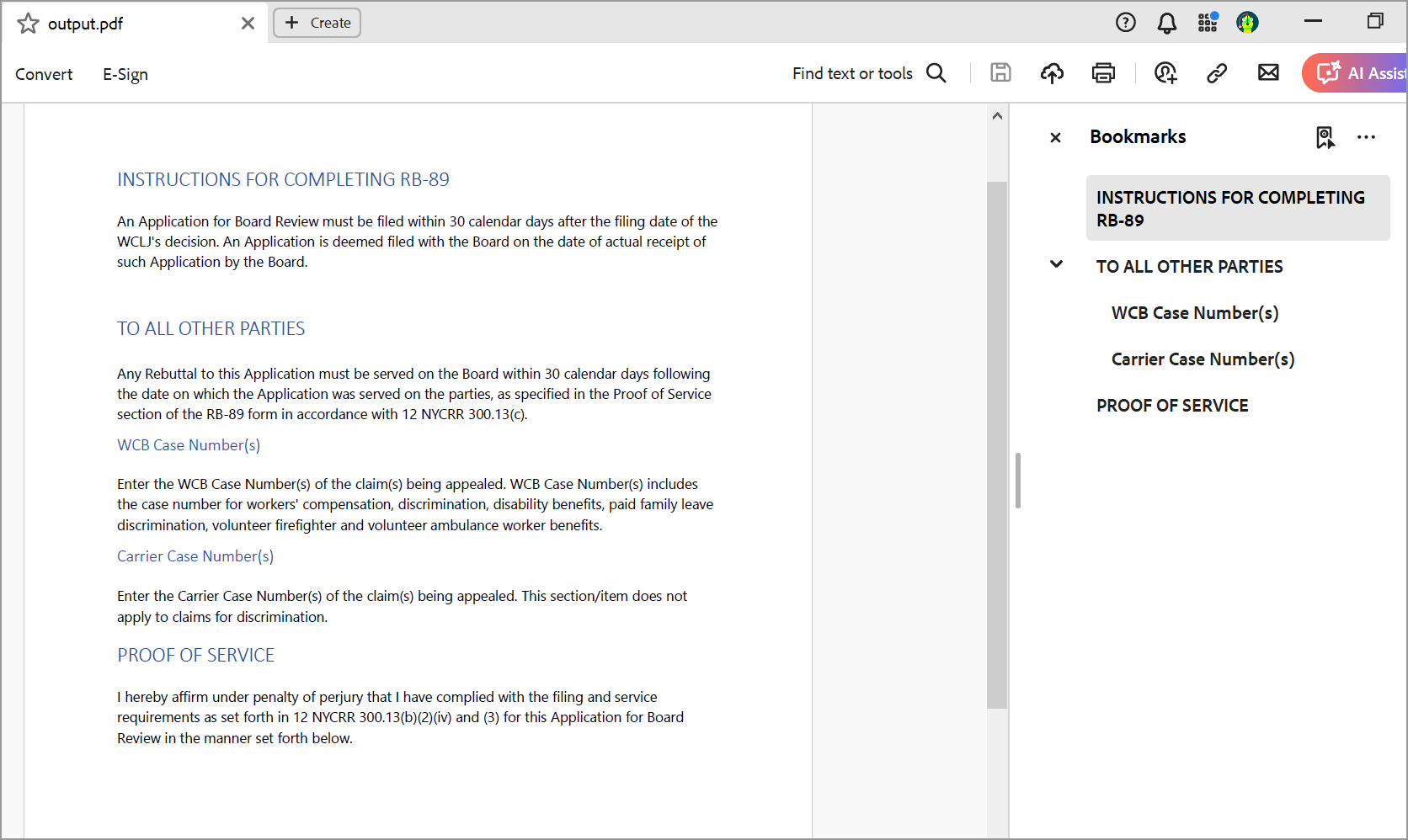
**break**;

}

**return** fontSize;

}

A complete working sample can be downloaded from [**create\_the\_automatic\_Bookmarking.zip**](https://www.syncfusion.com/downloads/support/directtrac/general/ze/Automatic_Bookmarking_in_PDF151005654.zip)

By executing the program, you will get the PDF document as follows.

Take a moment to peruse the [**documentation**](https://help.syncfusion.com/flutter/pdf/working-with-bookmarks), where you can find other options like [**adding**](https://help.syncfusion.com/flutter/pdf/working-with-bookmarks#adding-bookmarks-to-a-pdf), [**Inserting**](https://help.syncfusion.com/flutter/pdf/working-with-bookmarks#inserting-bookmarks-in-an-existing-pdf), [**removing**](https://help.syncfusion.com/flutter/pdf/working-with-bookmarks#removing-bookmarks-from-an-existing-pdf), and [**modifying bookmarks**](https://help.syncfusion.com/flutter/pdf/working-with-bookmarks#modifying-bookmarks-in-an-existing-pdf) in an existing PDF document and features like named destination, [**interactive annotations**](https://help.syncfusion.com/flutter/pdf/working-with-annotations), and [**insert hyperlink**](https://help.syncfusion.com/flutter/pdf/working-with-hyperlinks) to PDF with code examples.

Refer [**here**](https://www.syncfusion.com/document-processing/pdf-framework/net)to explore the rich set of Syncfusion Essential® PDF features.