***Project* tasks:**

Implement a word processor which allows the user to perform the following actions through a GUI (Graphical User Interface):

* On a page, write multiple paragraphs.
* Create a file with multiple pages.
* Store files in a hierarchy of folders.
* Open an existing file for modification.
* Save a file.
* Save a file as another file with a different name.
* Find and replace a word.
* Count number of words in a file.
* Bold, italicize, and underline text, as well as change the font size.
* Apply format painter.
* Insert text to a randomly selected location on the page.
* Cut or copy text from a randomly selected location on the page as well as paste the cut or copied text to a randomly selected location on the page.
* When a line is filled with text, the cursor will automatically move to the next line.
* Create headers, footers and page numbers.

1. **Abstract:**

For this project, our team decided to develop a word processor as a *desktop application*. To achieve this, we opted to utilize *Windows Form Application.NET Framework*, and implement it using the *C# programming language*. We aim to create a user-friendly and efficient word processor that meets the needs of our project requirements.

Top of Form

1. **Application Details**

:

* 1. **GUI (Graphical User Interface):**

The application is devided mainly into three main zones toolsbar-Shortcuts-Text\_space (as seen in the figure below):

A screenshot of a computer

Description automatically generated

* 1. **The required tasks:**
     1. **Task 1 | On a page, write multiple paragraphs:**
* **Task 1 | The output:**

To begin writing your own paragraphs, the first step is to navigate to **File** and select **AddPage**. This action will generate a new page, allowing you to start typing (as seen in the figure below).

A screenshot of a computer

Description automatically generated

* **Task 1 | The code:**

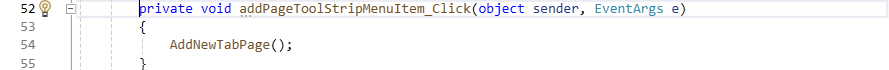
For this task, our initial step involves coding a function designed to facilitate the addition of a new page. This function is named AddNewTabPage() (as seen in the figure below).

A tab page is generated and docked according to the parent form, also, this is where eventhandlers are added, in addition, the label that counts the words is generated in this function and docked accordingly as shown below.

A computer screen shot of a program code

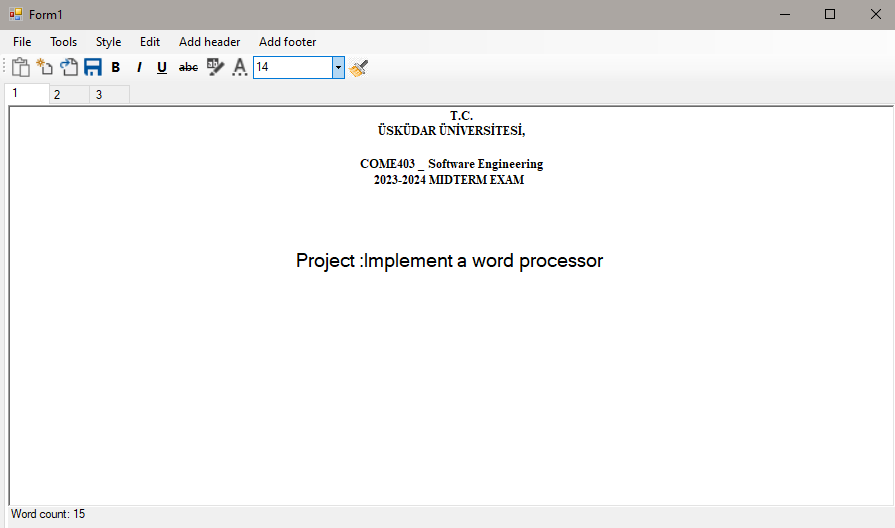
Description automatically generated

Next we integrate the AddNewTabPage() function into the addPageTool\_click event handler located within the StripMenu (as seen in the figure below).

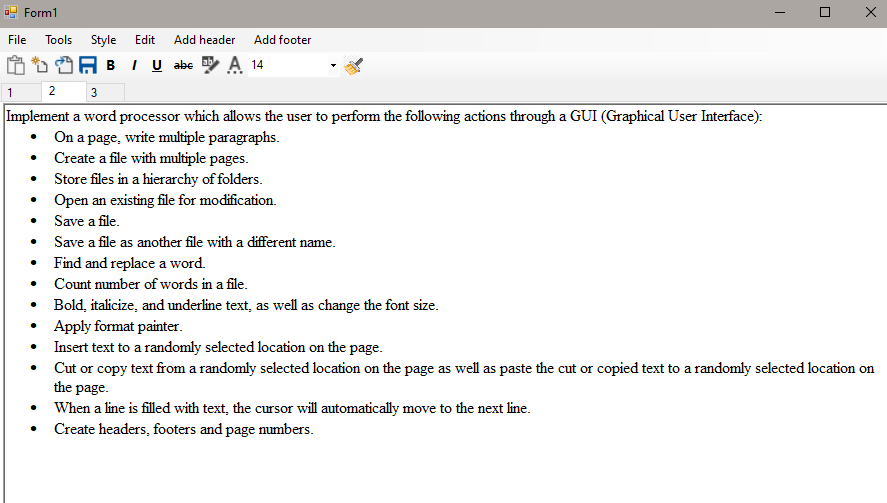


* + 1. **Task 2 | Create a file with multiple pages:**
* **Task 2 | The output:**

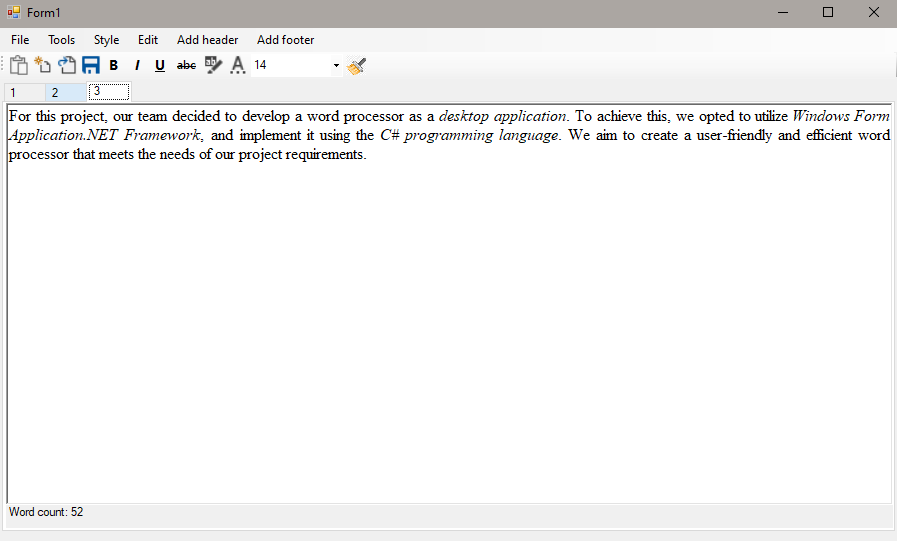
To initiate writing a file with multiple pages, simply navigate to the **File** menu and repeatedly select the **AddPage** option. Each time you perform this action, a new page will be generated, (as seen in the figure below).



Page 1



Page 2



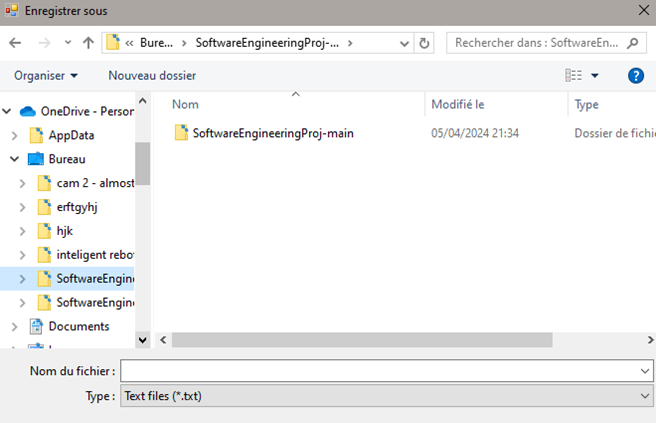
Page 3

* **Task 2 | The code:**

For this task, we use the same code as in task 1.

* + 1. **Task 3 | Store files in a hierarchy of folders:**
* **Task 3 | The output:**

To save your text, navigate to the **File** menu and choose the **Save** or **Save as** option. Select the desired *location* with hierarchy (as seen in the figure below).



Hierarchy folders

* **Task 3 | The code:**

The following points will provide an explanation of the code.

* + 1. **Task 4 | Open an existing file for modification:**
* **Task 4 | The output:**

An Open file function is implemented using a button in the menu strip, the code is shown below. The button opens a dialog that makes us choose a file similar to task 3 shown above, we made sure that all text files in a folder propagate to the menu and showing a multipage layout as requested in task 2.

A screenshot of a computer

Description automatically generated

* **Task 4 | The code:**

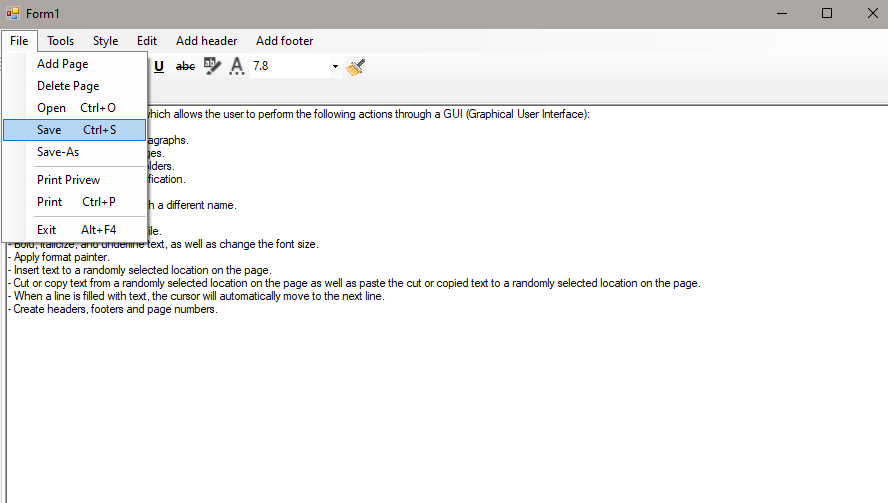
**A computer screen shot of a program

Description automatically generated**

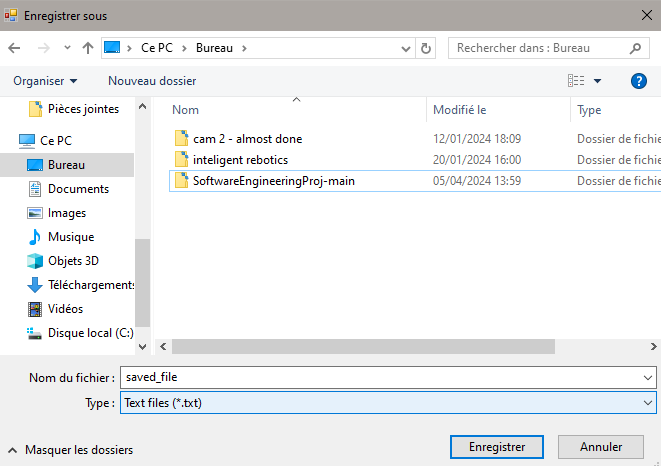
The code opens a folder browser dialog and lets us chose a folder, the text files in the folder are displayed as pages.

* + 1. **Task 5 | Save a file:**
* **Task 5 | The output:**

To save your text, navigate to the **File** menu and choose the **Save** option. Select the desired *location* where you wish to save the file and provide a *name*; a file with the *".txt"* extension will be created accordingly (as seen in the figure below).



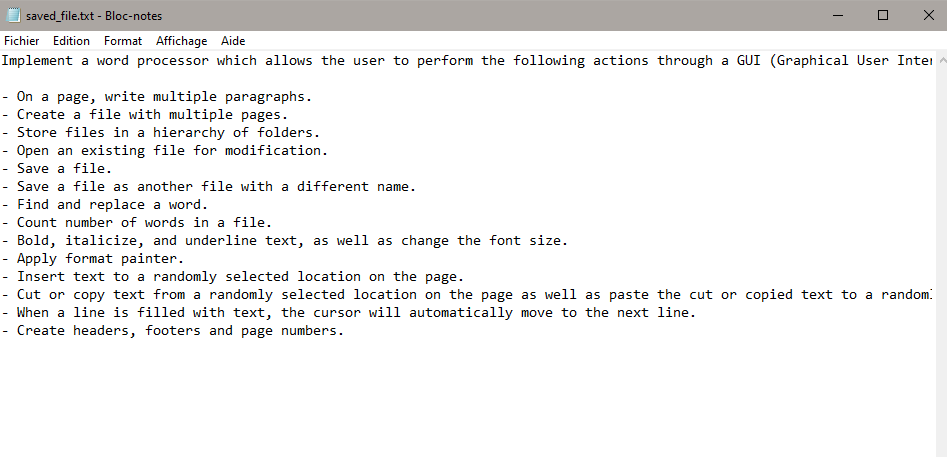
Select Save option



Location

Name

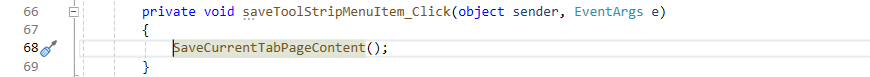
* **Task 5 | The code:**



The saved file

* **Task 5 | The code:**

To save text, we utilize a custom method named "SaveCurrentTabPageContent()" designed to save the content of the presently selected TabPage within a TabControl (as

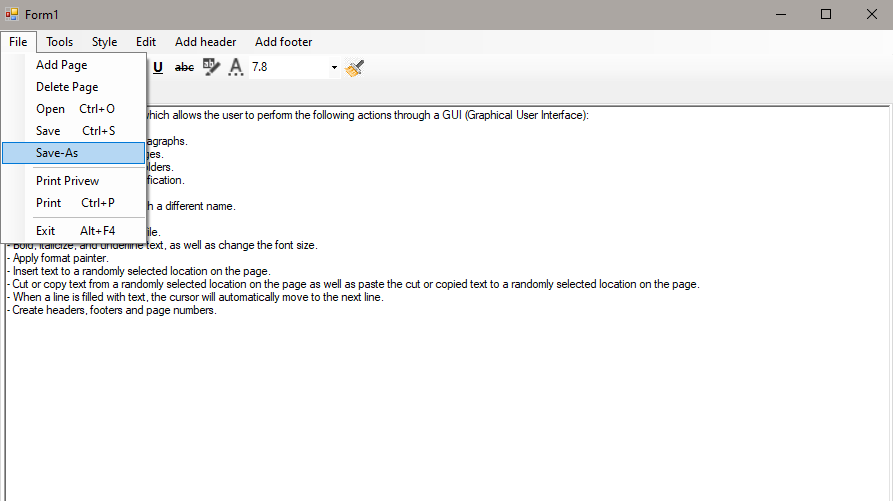
 seen in the figure below).

* + 1. **Task 6 | Save a file as another file with a different name:**
* **Task 6 | The output:**

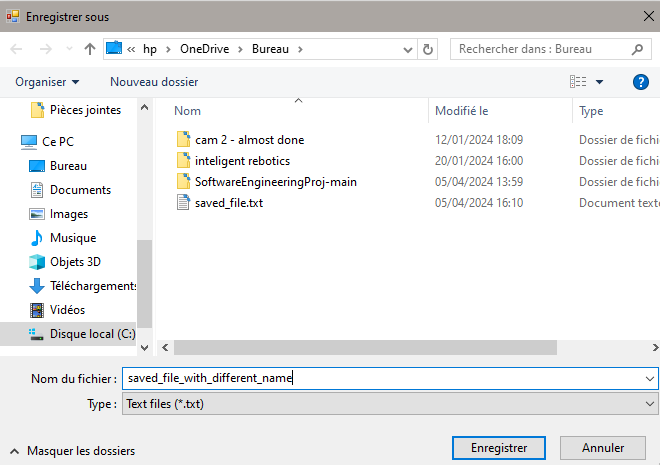
To save your text in other file in different name, navigate to the **File** menu and choose the **Save as** option. Select the desired *location* where you wish to save the file and provide *an other* *name*; a file with the *".txt"* extension will be created accordingly(as seen in the figure below).

A screenshot of a computer

Description automatically generated



Select Save as option



Different Name

Location

The new saved file

* **Task 6 | The code:**

For this task, this code displays a folder browser dialog to let the user choose a folder. If the user selects a folder and confirms, it saves the content of the currently selected TabPage using a custom method called SaveCurrentTabPageContent() (as seen in the figure below).

A screen shot of a computer code

Description automatically generated

* + 1. **Task 7 | Find and replace a word:**
* **Task 7 | The output:**

For this task, we the menu item **tools**, then hover to **find word,** when we press find word a dialog is prompted and the user is asked to enter the text that they want to replace, and the replacement text, (as seen in the figure below).

A screenshot of a computer

Description automatically generated

* **Task 7 | The code:**

**A screen shot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generated**

**A computer screen shot of a program code

Description automatically generated**

As shown above, we have a code for the prompt dialog, and we use the .**Replace** function to replace old text with desired text.

* + 1. **Task 8 | Count number of words in a file:**
* **Task 8 | The output:**

A screenshot of a computer

Description automatically generatedWhen a new tab is created and typing begins, the word count is automatically calculated and the number is shown at the bottom. the counter is applied to all pages.

The number of words

* **Task 8 | The code:**

For word counting, we implement a function named UpdateWordCount(). This function calculates the word count of the text within the RichTextBox and then updates the text of a Label to display the word count, the label is generated for any page we add using the add page button (as seen in the figure below).

**A screen shot of a computer code

Description automatically generated**

**A computer screen shot of a program code

Description automatically generated**

* + 1. **Task 9 | Bold, italicize, and underline text, as well as change the font size.**
* **Task 9 | The output:**

For this task, to style your text, you'll need to select the "Style" option. From there, you can choose the font, size, and apply formatting such as bold, italicize, and underline. Additionally, you can directly apply these styles by right-clicking on the corresponding icon-buttons (as seen in the figure below).

A screenshot of a computer

Description automatically generated

Style box

Icon buttons

* **Task 9 | The code:**

On this task , the code in *styleTool\_click* event handler located within the *StripMenu* opens a *font dialog* for the user to select a font. Then, it retrieves the *RichTextBox* control from the currently selected TabPage. After that, it sets the font of the selected text within the *RichTextBox* to match the font chosen by the user.

A screenshot of a computer

Description automatically generated

For icon-buttons , we utilize the custom method ToggleStyle function to activate features such as bold, italic, underline, and cross(as seen in the figure below). Top of Form

A screenshot of a computer program

Description automatically generated

The code for *highlight function* and *font\_color function* opens a color dialog for the user to select a color. If the user chooses a color and clicks "OK," it sets the background color of the selected text within a *RichTextBox* to the chosen color and sets the text color to the default text color of the *RichTextBox* (as seen in the figure below).

A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

* + 1. **Task 10 | Apply format painter**
* **Task 10 | The output:**

A format painter that copies the style and format of a highlighted text and applies it to a chosen text is applied as in the figures below.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Task 10 | The Code:**

A screen shot of a computer program

Description automatically generated

Applies the font style, color, and highlight from the current text to the new text by storing old customizations in variables and applies them to the highlighted text

* + 1. **Task 11 | Insert text to a randomly selected location on the page.**
* **Task 11 | The output:**

After adding a new page, we have the capability to insert text at a randomly selected location on that page (as seen in the figure below).

**A screenshot of a computer

Description automatically generated**

* **Task 11 | The Code:**

For this part the richtextbox is automatically able to have text inserted into a random location.

* + 1. **Task 12 | Cut or copy text.**
* **Task 12 | The output:**

To copy, cut, or paste text, you can either select the corresponding tools from the "Edit" menu in the MenuStrip or use the shortcuts Ctrl+C, Ctrl+X, and Ctrl+V, respectively (as seen in the figure below).

**A screenshot of a computer

Description automatically generated**

* **Task 12 | The code:**

These methods ensure that the required operations (cut/copy/paste) are performed on the text within the RichTextBox control on the currently selected TabPage (as seen in the figure below).

**A screenshot of a computer program

Description automatically generated**

* + 1. **Task 13 | When a line is filled with text, the** **cursor will automatically move to the next line.**
* **Task 13 | The output:**

**A screenshot of a computer

Description automatically generated**After adding a new page, we have the capability to move to the next line when line is filed with text by pressing the format > Text wrap, the feature was added later. (as seen in the figure below).

**A screenshot of a computer

Description automatically generated**

* **Task 13 | The Code:**

A screen shot of a computer program

Description automatically generated

The text wrap function shown above disables or enables the word wrap functionality of the richtextbox where the cursor at the end of the line goes to the beginning of the next line depending on the choice of the user.

* + 1. **Task 14 | Create headers, footers and page numbers.**
* **Task 14 | The output:**

An option for adding a header and another option for adding a footer are added as menu items, upon pressing any of them, a dialog is prompted, the text written in the dialog will be appended as a footer or a header along with a horizontal line depending on the choice.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* **Task 14 | The code:**

**A computer screen shot of a program

Description automatically generated**

**A screen shot of a computer program

Description automatically generated**

The code is explained in the comments.

A screen shot of a computer program

Description automatically generated