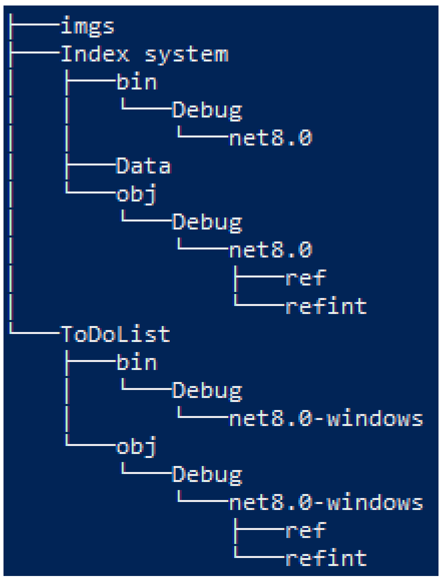
U16A2

file structure



```
├───imgs
├───Index system
│   ├───bin
│   │   └───Debug
│   │       └───net8.0
│   ├───Data
│   └───obj
│       └───Debug
│           └───net8.0
│               ├───ref
│               └───refint
└───ToDoList
    ├───bin
    │   └───Debug
    │       └───net8.0-windows
    └───obj
        └───Debug
            └───net8.0-windows
                ├───ref
                └───refint
```

problem 1

A Todo list.

Pseudocode for XAML (UI Layout)

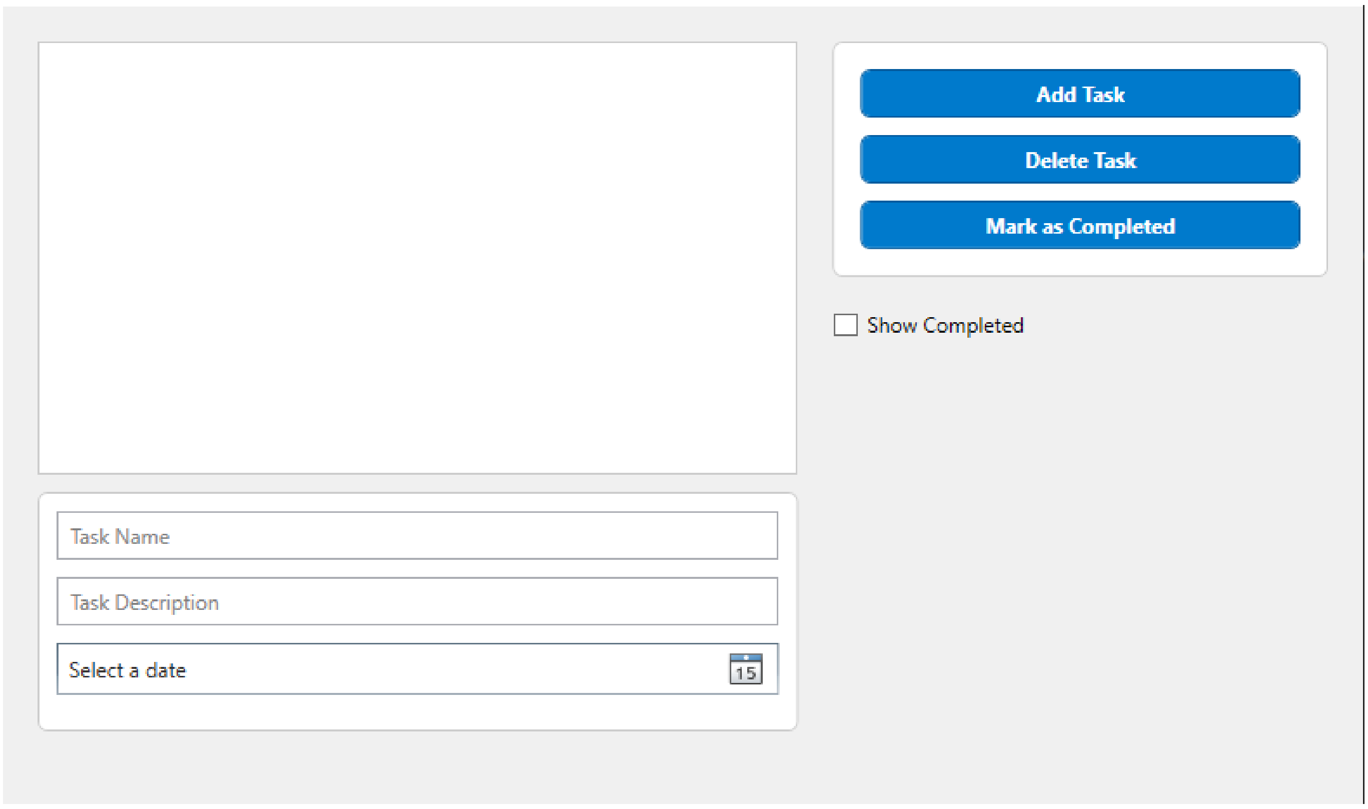
```
Window
    Set title to "To-Do List"
    Set dimensions and background color

    Define styles
        ButtonStyle
            Set margin, padding, background, foreground, font weight, border
brush, and border thickness
            Define template with border and content presenter

        PlaceholderTextBoxStyle
            Set foreground to gray
            Define event handlers for focus events to manage placeholder text

    Main Grid
        Define two columns (left 3 parts, right 2 parts)

        Left Column
            StackPanel
                ListBox for displaying tasks

                Border
                    StackPanel
                        TextBox for task title with placeholder style
                        TextBox for task description with placeholder style
                        DatePicker for due date

        Right Column
            StackPanel
                Border
                    StackPanel
                        Button to add task
                        Button to delete task
                        Button to mark task as completed

                CheckBox to show completed tasks
```

Pseudocode for C# (Code-Behind)

```
Class MainWindow
    Define Tasks as an observable collection of TaskItem

    Constructor
        Initialize components
        Initialize Tasks collection
        Bind TaskList to Tasks collection

    Method AddTask_Click
        Create new task with input values
        Add new task to Tasks collection
        Clear input fields

    Method DeleteTask_Click
        If a task is selected in TaskList
            Remove selected task from Tasks collection

    Method MarkAsComplete_Click
        If a task is selected in TaskList
            Mark selected task as completed
            Refresh TaskList

    Method ShowCompletedTasks_Checked
        If checkbox is checked
            Show only completed tasks in TaskList
        Else
            Show only incomplete tasks in TaskList

    Method ClearInputFields
        Reset input fields to default placeholder text

    Method RemovePlaceholderText
        If textbox has placeholder text
            Clear text and change foreground color

    Method AddPlaceholderText
```

```
        If textbox is empty
            Set placeholder text and change foreground color

Class TaskItem
    Define properties for title, description, due date, and completion status

    Method ToString
        Return string representation of task item
```

display layout



problem 2

The college Library needs to add all its books to a new index system.

Pseudocode

```
// Define constants for input and output file paths
inputFile = "Data/book_data.csv"
outputFile = "Data/UPDATED_book_data.csv"

// Read all lines from the input CSV file
lines = ReadAllLinesFromFile(inputFile)

// Initialize a StringBuilder to store modified CSV data
sb = StringBuilder()

// Loop through each line in the input CSV file
foreach line in lines:
    // Split the line into columns using comma as delimiter
    columns = SplitLineIntoColumns(line)

    // Generate a hash based on all columns
    hash = GenerateHash(columns)

    // Append the original line with appended hash to StringBuilder
    sb.AppendLine(line + "," + hash)

// Write the modified CSV data to the output file
WriteToFile(outputFile, sb.ToString())

// Output success message
Print("CSV file processed successfully!")

// Function to split a line into columns
function SplitLineIntoColumns(line):
    return Split(line, ',')

// Function to generate hash from array of strings
function GenerateHash(inputs):
    // Initialize MD5 hash algorithm
    md5 = InitializeMD5()

    // Concatenate all inputs into a single string
    concatenatedInput = Join(inputs, "")

    // Convert concatenated string to bytes
    inputBytes = ConvertStringToBytes(concatenatedInput)

    // Compute hash bytes using MD5
    hashBytes = md5.ComputeHash(inputBytes)

    // Convert hash bytes to hexadecimal string
    hashString = ConvertBytesToHexString(hashBytes)

    // Take the first 8 characters as the hash number
    return Substring(hashString, 0, 8)

// Function to initialize MD5 hash algorithm
function InitializeMD5():
    return MD5.Create()

// Function to convert string to bytes using UTF-8 encoding
function ConvertStringToBytes(input):
    return UTF8.GetBytes(input)

// Function to convert bytes to hexadecimal string
function ConvertBytesToHexString(bytes):
    sb = StringBuilder()
    for each byte in bytes:
        sb.Append(ByteToHexString(byte))
    return sb.ToString()
```

```
// Function to convert byte to hexadecimal string
function ByteToHexString(byte):
    return Format(byte, "X2")

// Function to write data to a file
function WriteToFile(filePath, data):
    WriteAllText(filePath, data)

// Function to read all lines from a file
function ReadAllLinesFromFile(filePath):
    return ReadAllLines(filePath)

// Function to print message to console
function Print(message):
    Console.WriteLine(message)
```

test plan

problem 1

Test	**Expected results**	**actual results**
click add task and see if it adds a task with the correct data	test - testing description Due 07/06/2024 - incomplete	
click Mark as completed to see if the status on the	test - testing description Due 07/06/2024 - Completed	
click delete task and see if the selected task is deleted	removes the selected task	
add 15 testing tasks	scroll bar added allowing scrolling	

Test	**Expected results**	**actual results**
click add task and see if it adds a task with the correct data	test - testing description Due 07/06/2024 - incomplete	test - testing description Due 07/06/2024 - incomplete
click Mark as completed to see if the status on the	test - testing description Due 07/06/2024 - Completed	test - testing description Due 07/06/2024 - Completed
click delete task and see if the selected task is deleted	removes the selected task	removes the selected task
add 15 testing tasks	scroll bar added allowing scrolling	scroll bar added allowing scrolling

problem 2

Test	Expected results	actual results
run the code to make sure it works	nothing breaks	
run the code the make sure the CSV file is found	the code should keep working	
run the code to see if it outputs a new CSV file	it should create a new CSV file	
open the new CSV file to make sure it has hash number next to it	the CSV file should have an extra column with the hashs in it	

Test	Expected results	actual results
run the code to make sure it works	nothing breaks	nothing broke
run the code the make sure the CSV file is found	the code should keep working	the code keeps running
run the code to see if it outputs a new CSV file	it should create a new CSV file	it created a new CSV file
open the new CSV file to make sure it has hash number next to it	the CSV file should have an extra column with the hashs in it	there is a new collum with hash numbers

feedback

Aiden Scowen: It seems like you have a good understanding of the problem and have provided a clear and detailed solution. The only problem I see is that for problem two the code seems to only take the first column and hash that so if the first column is the same it will generate the same hash even is everything else is different.

Adam Hurst: You seem to have met the requirements for the problem and have provided a good solution to their problems. The only thing I would say is that your delete task button on the todo list does not seem to work.

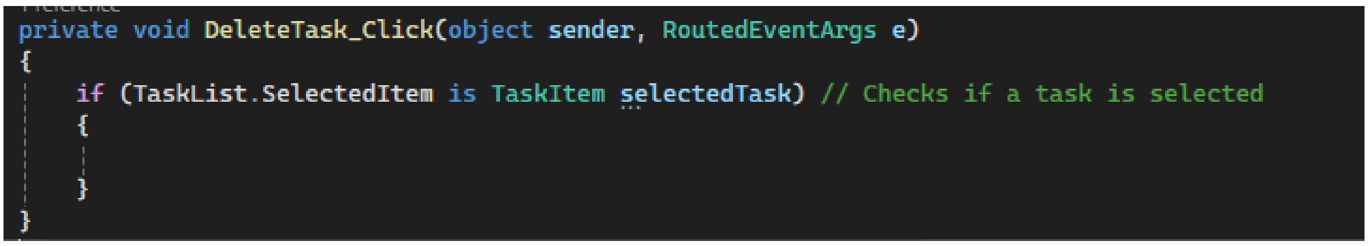
response

I shall look into the delete task button and fix the problem. I will also look into the hashing problem and see if I can change it to read all the columns instead of just the first one.

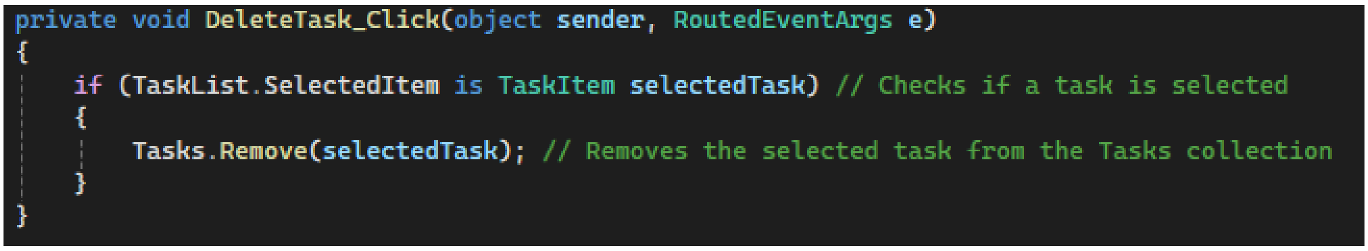
optimization

problem 1

I have forgot to add the code to actually delete the task from the list. I have added the code to delete the task from the list and it now works as expected.



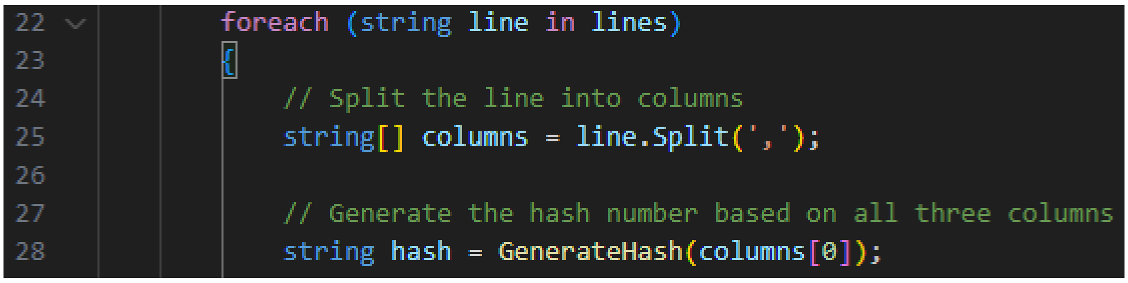
```
private void DeleteTask_Click(object sender, RoutedEventArgs e)
{
    if (TaskList.SelectedItem is TaskItem selectedTask) // Checks if a task is selected
    {

    }
}
```



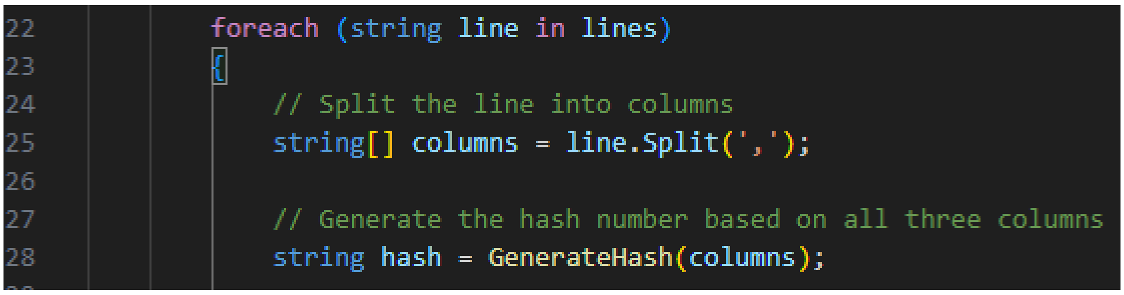
```
private void DeleteTask_Click(object sender, RoutedEventArgs e)
{
    if (TaskList.SelectedItem is TaskItem selectedTask) // Checks if a task is selected
    {
        Tasks.Remove(selectedTask); // Removes the selected task from the Tasks collection
    }
}
```

problem 2

I have changed it so the columns it uses to generate the hash is not set to 0 so it reads all the columns instead of just the first one.



```
foreach (string line in lines)
{
    // Split the line into columns
    string[] columns = line.Split(',');

    // Generate the hash number based on all three columns
    string hash = GenerateHash(columns[0]);
```



```
foreach (string line in lines)
{
    // Split the line into columns
    string[] columns = line.Split(',');

    // Generate the hash number based on all three columns
    string hash = GenerateHash(columns);
```

Justification

I believe that I have met the requirements for the problems and have provided a good solution to the problems.

For problem 1 the reason why I added all the buttons I have is because they are the basic needs for a to do list without making it to over complicated and hard to use. As for text boxes I wanted the users to be able to put detailed information in the task to make it easier to understand later on. The date picker is so the user can set a due date for the task if they need to. As for the show completed tasks I wanted feel like having the user be able to see what task are completed will allow them to focus on other tasks.

For problem 2. The reason I chose hashing as the way to index the books is because it's the easiest way to give unique numbers to each book. With hashing you can change the size of the hash number if your working with large amount of data. Another reason why hashing is the best for this solution is because it's almost impossible to generate the same hash number for two different data sets. This means that the library will not have to worry about two books having the same index number.

Evaluation

When it came to writing the code I have found it difficult to work with writing/reading CSV files as that was new for me. One of the hardest problems I faced was trying to get the code to make a new CSV file with all the original data plus having a new column with the hashed numbers. I think I have improved with making CSV files and I feel a lot more confidant in being able to make a better one next time.

I did have some problems UI layout as I have barely worked with XAML before. But it was not as hard as I thought it would be. Having to switch between the XAML and the C# code was a bit annoying but I got used to it after a while. I think I have improved with XAML and I feel a lot more confidant in being able to make one in shorter amount of time.