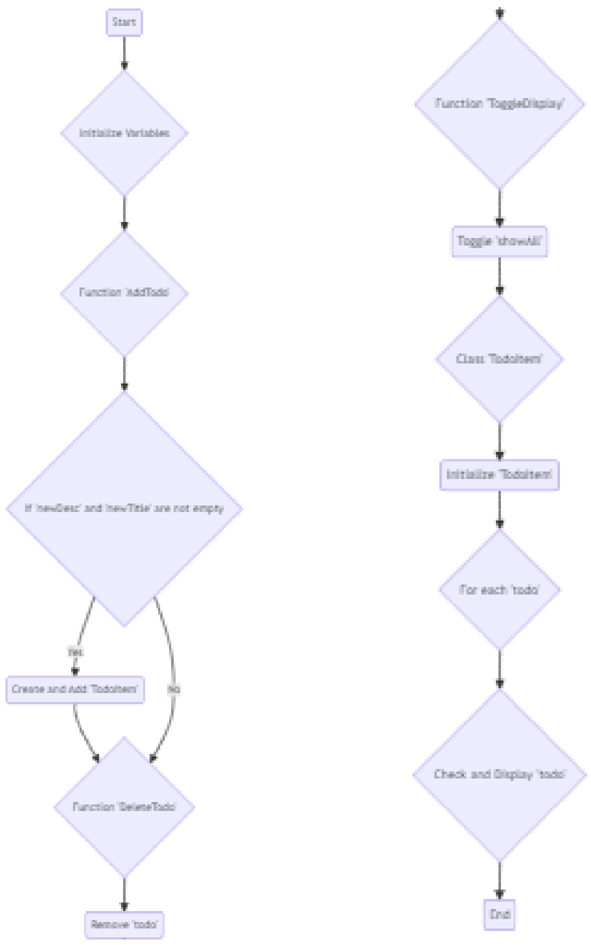
The Hashing Application

The Library Index System will read the details from a CSV file, create a unique index reference, and write the unique reference for each line to a new CSV file as a way to automate the process of indexing books in a college library. This method will ensure that every book has a unique identity and make the indexing process much more efficient. With file management, data processing and indexing system implementation, the challenge is averagely complex.

One of the constraints for this application is the fact that the data must be read and written by the programme in CSV format. This restriction may cause issues if the format of the CSV file changes or if the data contains errors (missing fields, unnecessary commas, etc.), which may need further errors correcting.



((Start)

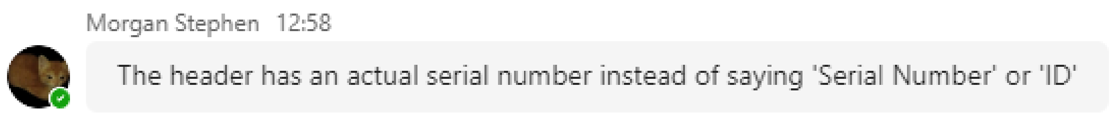
{Initialize Variables}
Function 'AddTodo'}
If 'newDesc' and 'newTitle' are not empty}
(Create and Add 'TodoItem')
{Function 'DeleteTodo'}

(Remove 'todo')
{Function 'ToggleDisplay'}
(Toggle 'showAll')
{Class 'TodoItem'}
Initialize 'TodoItem')
{For each 'todo'}
{Check and Display 'todo'}

(End))

Test Number	Steps	Description	Expected Results	Actual Results	Comments
1	Provide a sample CSV file with valid book details and run.	Reading Book Details from CSV	Serial number is added correctly.	Serial number is added correctly.	N/A
2	Check hashed file.	Generating Unique Index References	Serial numbers are unique.	Serial numbers are unique.	N/A
3	Provide an incorrect csv file and run app.	Error Handling	Hash not provided with no info.	Hash not provided with no info.	N/A

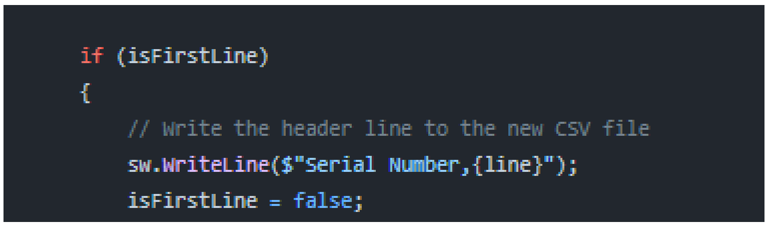
Feedback



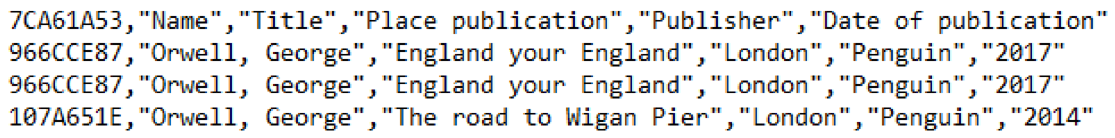
Morgan Stephen 12:58

The header has an actual serial number instead of saying 'Serial Number' or 'ID'

Before Optimizations:

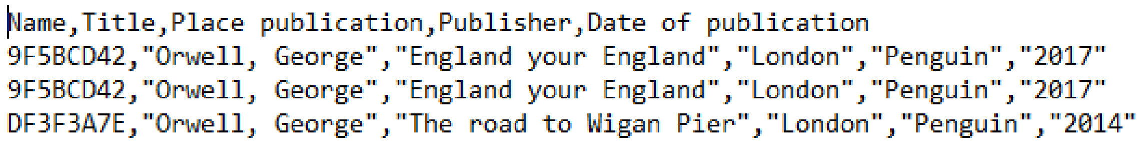


```
    if (isFirstLine)
    {
        // Write the header line to the new CSV file
        sw.WriteLine($"Serial Number,{line}");
        isFirstLine = false;
```

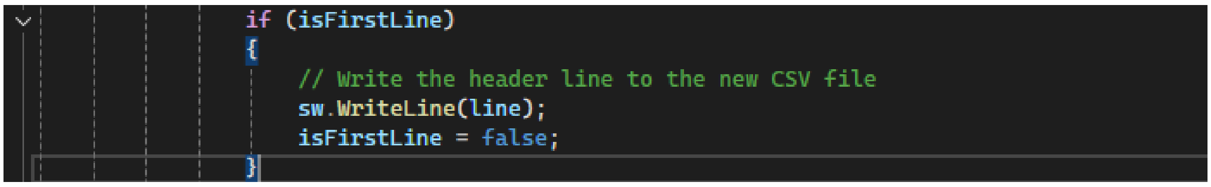


```
7CA61A53,"Name","Title","Place publication","Publisher","Date of publication"
966CCE87,"Orwell, George","England your England","London","Penguin","2017"
966CCE87,"Orwell, George","England your England","London","Penguin","2017"
107A651E,"Orwell, George","The road to Wigan Pier","London","Penguin","2014"
```

After:



```
Name,Title,Place publication,Publisher,Date of publication
9F5BCD42,"Orwell, George","England your England","London","Penguin","2017"
9F5BCD42,"Orwell, George","England your England","London","Penguin","2017"
DF3F3A7E,"Orwell, George","The road to Wigan Pier","London","Penguin","2014"
```



```
if (isFirstLine)
{
    // Write the header line to the new CSV file
    sw.WriteLine(line);
    isFirstLine = false;
}
```

Before optimization, the hashed_books would contain the first line of the CSV file that is meant to show the details of the below information with it's own hash. In order to fix this, I used an if statement to single out the first line and print it without the hash by deleting the line that adds it's own $"Serial Number,.

Data Dictionary

Variable	*Type*	*Description*
hash	int	Stores the hash code of the input string in the GenerateSerialNumber.
path	string	The file path to the input CSV file (books.csv).
outputPath	string	The file path to the output CSV file (hashed_books.csv).
isFirstLine	bool	A flag to indicate whether the current line being processed is the first line of the CSV file.
fields	string	An array of strings representing the fields of the current line being read from the CSV file.
line	string	The strings of the current line's fields that are joined by commas.
serialNumber	string	The generated serial number for the current line.
quotedLine	string	The current line's fields including the serial number at the beginning.

Review against client requirements

The client has requested that I create an application which will automatically read book details from a CSV file and output a unique index reference to a new CSV. The following are the criteria:

- **Read book details from a stored CSV File**

The code reads the CSV file using TextFieldParser from the Microsoft.VisualBasic.FileIO namespace.

- **Generating unique index reference**

The code uses an interface ISerialNumberGenerator to define a method for generating serial numbers. The HashSerialNumberGenerator class generates serial numbers by converting the input into a hexadecimal string that is 8 characters long.

- **Writing to a new CSV file**

The outputPath is set to output the new hashed books to a file called "hashed_books.csv", and uses "StreamWriter" in order to write the new data to this file.

- **Processing each line of the CSV**

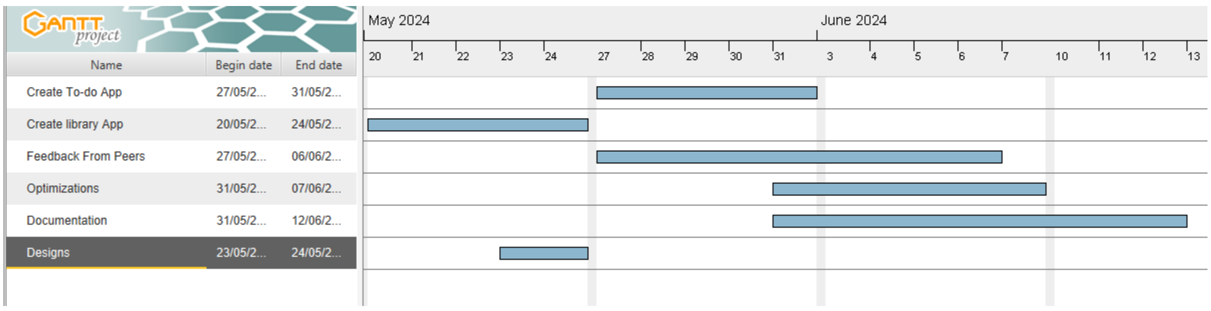
The code reads each line of the CSV file, generates a serial number, and writes the new line with the serial number to the output file as demonstrated by the flowchart.

- Demonstrate encapsulation

The code defines classes "HashSerialNumberGenerator" "Program" and uses objects to encapsulate functionality.

Planning and Managing My Time

My progress has been greatly enhanced by setting deadlines and goals. I divided more complex tasks into smaller jobs with specific due dates. This method enabled me to efficiently manage my workload. I was also sure to review my work with peers throughout the project and gain feedback. This allowed me to constantly optimize the code and plan using flowcharts to meet client requirements more effectively. I decided to create a Gantt chart in order to better visualize my deadlines:



Name	Begin date	End date
Create To-do App	27/05/2...	31/05/2...
Create library App	20/05/2...	24/05/2...
Feedback From Peers	27/05/2...	06/06/2...
Optimizations	31/05/2...	07/06/2...
Documentation	31/05/2...	12/06/2...
Designs	23/05/2...	24/05/2...

Evaluation

My code efficiently demonstrates how to process CSV data (books.csv) in a structured way and create a unique serial number for each item before writing it to a different CSV file (hashed_books.csv). The first step is creating an interface called ISerialNumberGenerator and using HashSerialNumberGenerator to implement it. This generator uses the hash codes of the input texts to translate them into hexadecimal serial numbers.

This design decision allows for flexibility by enabling the introduction of different serial number generation algorithms in the future without requiring significant code changes. However, it would be easier to read and maintain in the future if the code had better documentation that included more comments outlining design choices. Overall, the code effectively meets client requirements and achieves its functional objectives.

Methodology Evaluation

Within this project I utilized the Agile methodology, which involves breaking the project into phases and involves a cycle of planning, execution and evaluation (Source: Atlassian)

Plan: I understood the client's needs, which wanted me to create a unique index reference for each book within a CSV file. I then deconstructed the requirements into tasks that needed to be accomplished inside a Gantt chart that I created.

Design: Using the client's specifications for the application, I created a functional code design that includes comments for easy understanding and ensures functionality.

Develop: I efficiently developed the code for the application following the Gantt chart's strategic planning.

Test: To ensure that every implemented feature complies with client requirements, I ran functional tests. Reading Book Details from CSV, Generating Unique Index References and Error Handling were all tested.

Deploy: After testing the application, I successfully deployed the application to GitHub.

Review: I received feedback from peers multiple times and detailed how I used this feedback for optimization and evaluated the whole project.