Certﬁcation of Project completion

# Java\_Creating a new File-Saving it in new location-Counting characters.

September, 2023

**BY**

* Anisiuba Mmesoma Benedicta (Student1427035)
* Ege Obetta Amarachi (Student143711)
* Ochinanwata Majesty (Student1429696)
* Kanu Daniel Chibuzor (Student1434706)
* Olisaemeka Sophia (Student1434703)
* Agu Martha (Student1434710)

# Table of content

[Acknowledgement](#_Acknowledgement) 3

[Problem Deﬁnition](#_Problem Definition) 4

[Project Speciﬁcation](#_Project Specification) 6

[E-Project Analysis](#analysis) 8

[E-Project Design](#_Project Design) 9

[Data Flow Diagram](#design) 9

[Flowcharts](#flowchart) 10

[User guide / Developer guide](#_User / Developer Guide) 11

[Source code](#_Source Code) 13

[Task sheet](#_Task Sheet) 15

[Checklist of validation](#_Checklist of Validation) 16 [Submission checklist](#_Submission Checklist) 17

# Acknowledgement

To our remarkable team,

This project has been a journey filled with creativity, collaboration, and shared passion. We want to express our deepest gratitude to each and every one of you for your tireless dedication and unwavering support. Your unique talents and unwavering commitment have not only contributed to the success of this endeavor but have also made it an enriching experience.

Through the challenges and triumphs, we've grown together as a team, and the memories we've created will forever hold a special place in our hearts. To our mentors and advisors, your guidance has been our compass, and we are immensely thankful for your wisdom.

To our friends and families, your patience and encouragement sustained us during late nights and countless revisions. Your belief in us meant the world.

This project is not just a culmination of our hard work; it's a testament to the power of collaboration, friendship, and shared dreams. We couldn't have asked for a better team. Thank you for making this journey truly exceptional.

# Problem Definition

--

Create a Java program that reads the content of two files, combines them into a third file, and displays the merged content on the console. Additionally, calculate and display the character count of the merged file. Save this character count in a separate file. Ensure proper code documentation.

--

**Problem Overview:**

**You are tasked with developing a Java program that performs the following tasks:**

* Read and display the content of two input files on the console.
* Combine the content of these files and save it into a third file.
* Display the merged content on the console.
* Calculate and display the number of characters in the merged file.
* Save the character count in a separate file.
* Ensure that the code is well-documented with comments explaining the logic.
* Prepare a complete project report, including a synopsis, code, and documentation.

Example:

If File A.txt contains "ABCDEFGHIJKLM" and File B.txt contains "NOPQRSTUVWXYZ," the merged file should contain "ABCDEFGHIJKLMNOPQRSTUVWXYZ."

**Output Requirements:**

* The program should provide console output of the content of the two input files.
* It should create a third file containing the merged content and display it on the console.
* The program should calculate and display the character count of the merged file.
* The character count should be saved in a separate file.
* Proper code documentation should be maintained.

**Standards Plan:**

* Ensure that every code block is well-commented for clarity.
* Provide clear explanations and documentation for the program's logic.
* Create a comprehensive project report that includes a synopsis, code, and detailed documentation.

# Project Specification

**Functional Requirements:**

* Input File Processing: The program should read the content of two input files provided by the user.
* Content Display: Display the content of both input files on the console screen to validate the input.
* Merging Files: Combine the contents of the two input files into a third file.
* Merged File Display: Display the content of the merged file on the console screen.
* Character Counting: Calculate and display the number of characters present in the merged file.
* Character Count Saving: Save the character squared count in a separate file.

**Non-Functional Requirements:**

**-- Documentation:**

Proper code comments and documentation should be maintained throughout the project to ensure clarity and understanding.

**-- Project Report:**

Prepare a comprehensive project report containing:

* Certificate of Completion.
* Table of Contents.
* Problem Definition.
* Algorithms..
* Task sheet.
* Project Review and Monitoring Report.
* Final Check List.

**Hardware/Software Requirements:**

**-- Hardware:**

A computer system with at least a Pentium 166 processor.

Minimum 128 Megabytes of RAM.

**-- Software:**

The program can be developed using one or a combination of the following programming languages:

Java / J2EE / .NET / C / C++

Text editor (e.g., Notepad).

# E-Project Analysis

The project, known as the "File Concatenation and Character Counting Program," is designed to address fundamental tasks in file manipulation and data processing. Its core objective is to create a Java-based software tool that streamlines the merging of content from two input files into a designated third file while concurrently computing the character count of the merged output. This multifaceted program caters to a diverse audience, including developers, programmers, and educational institutions seeking an accessible yet robust utility. With Java as its programming language, it ensures compatibility across various Java Integrated Development Environments (IDEs) or even plain text editors like Notepad.

**-- Key Features and Functionality:**

This versatile program offers a range of essential features, including input file processing, transparent content display, efficient file merging, user-friendly presentation of merged content, and accurate character counting capabilities. Its clear and well-documented code comments uphold high coding standards and enhance overall code quality. It's worth noting that the program operates under the assumption of structured input files, exclusively handling plain text files. For execution, it relies on the Java Runtime Environment (JRE), ensuring broad accessibility.

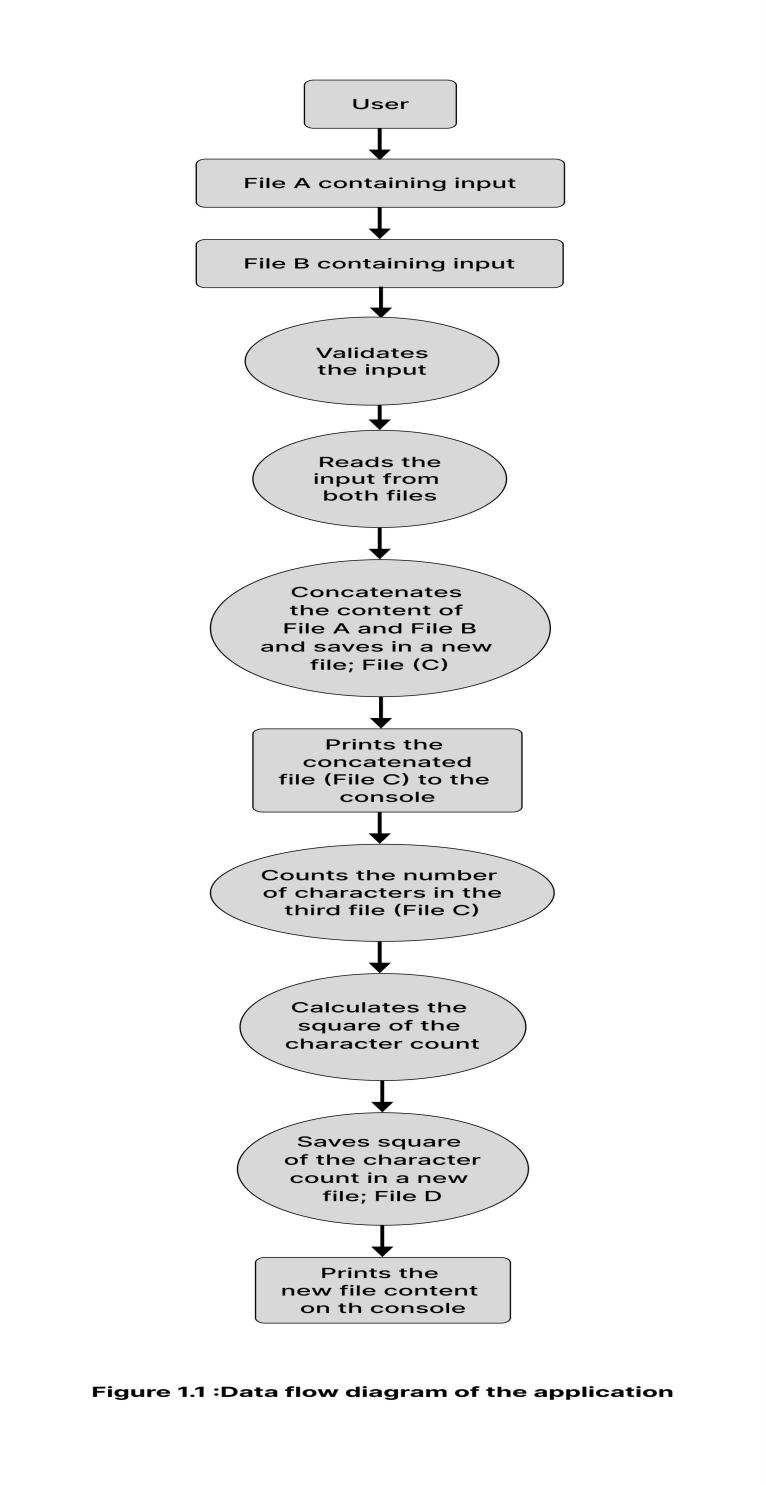
**-- Challenges and Success Metrics:**

While the project foresees potential challenges like managing larger input files or accommodating variations in file access permissions across different systems, its ultimate success hinges on delivering reliable functionality, strict adherence to coding standards, and the creation of comprehensive documentation. This project analysis underscores its significance as a practical tool for file manipulation, an educational resource for teaching file handling concepts, and a testament to the capabilities of Java-based programming.

# Project Design

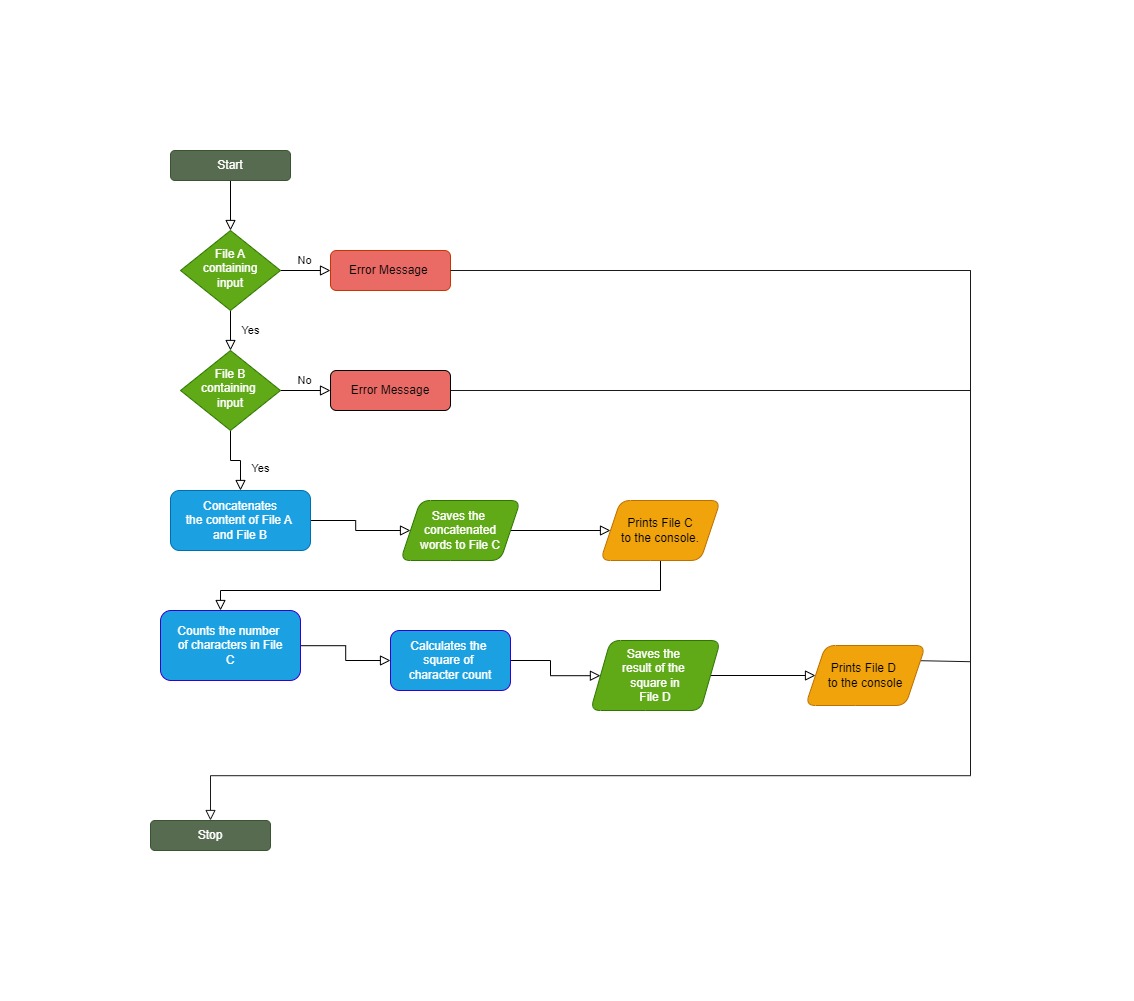
**Data Flow Diagram:**

This DFD represents the program's data flow process, illustrating how user-provided data evolves within the system.



**Flowchart Diagram:**

This Flowchart outlines the sequential steps and decision points within the program's workflow.



# User / Developer Guide

**User Guide :**

* Ensure you have the Java Runtime Environment (JRE) installed on your computer.
* Download the program and extract the downloaded file to a folder of your choice.
* Place the two files you want to concatenate in the same folder as the program and name them "fileA" and "fileB" respectively.
* Alternatively, paste the content of the first file you want concatenated and counted in "fileA" and the content of the other file in "fileB."
* Open the program by double-clicking the executable file.
* The program will automatically detect "fileA" and "fileB" in the same folder.
* Follow on-screen instructions to proceed.
* The program will concatenate the data from the selected files and the merged data will be saved in a new output file.

That's it! You've successfully used the program to concatenate data from two files.

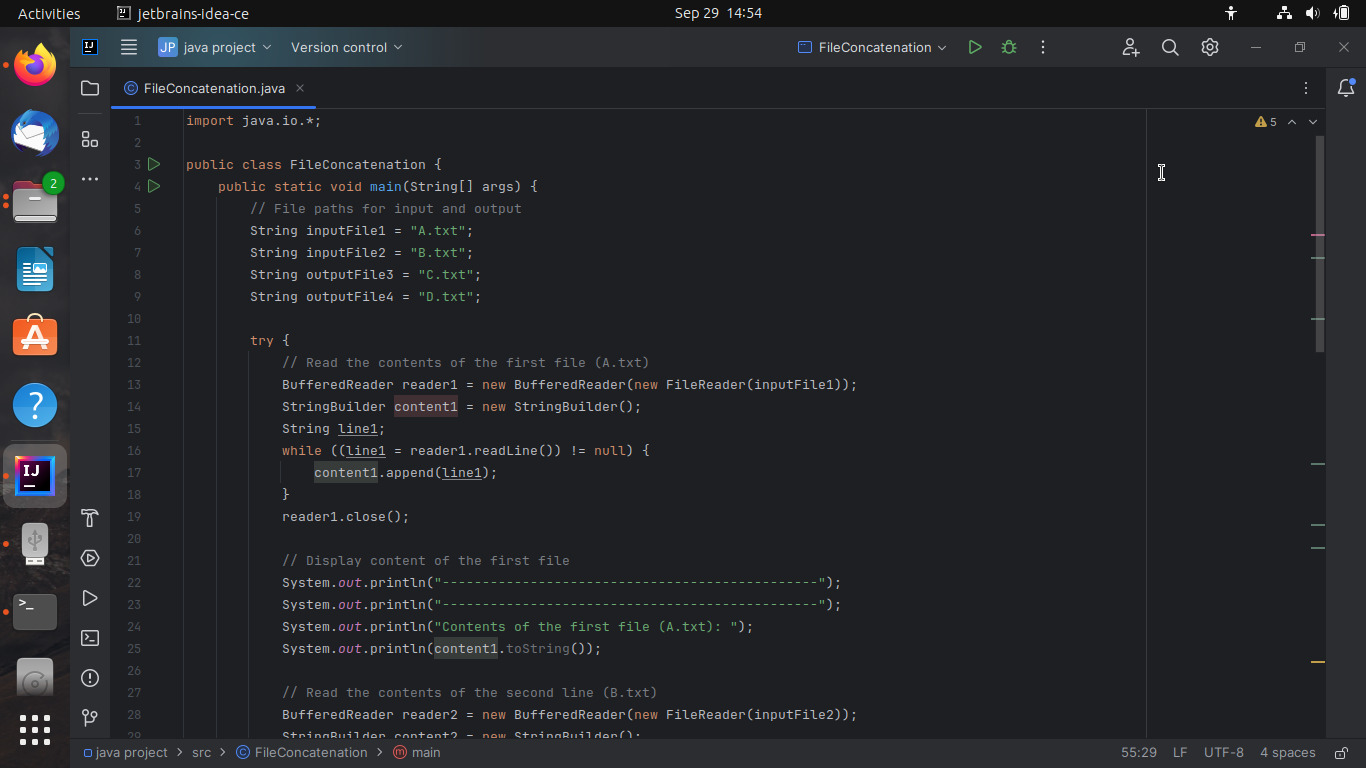
**Developer Guide :**

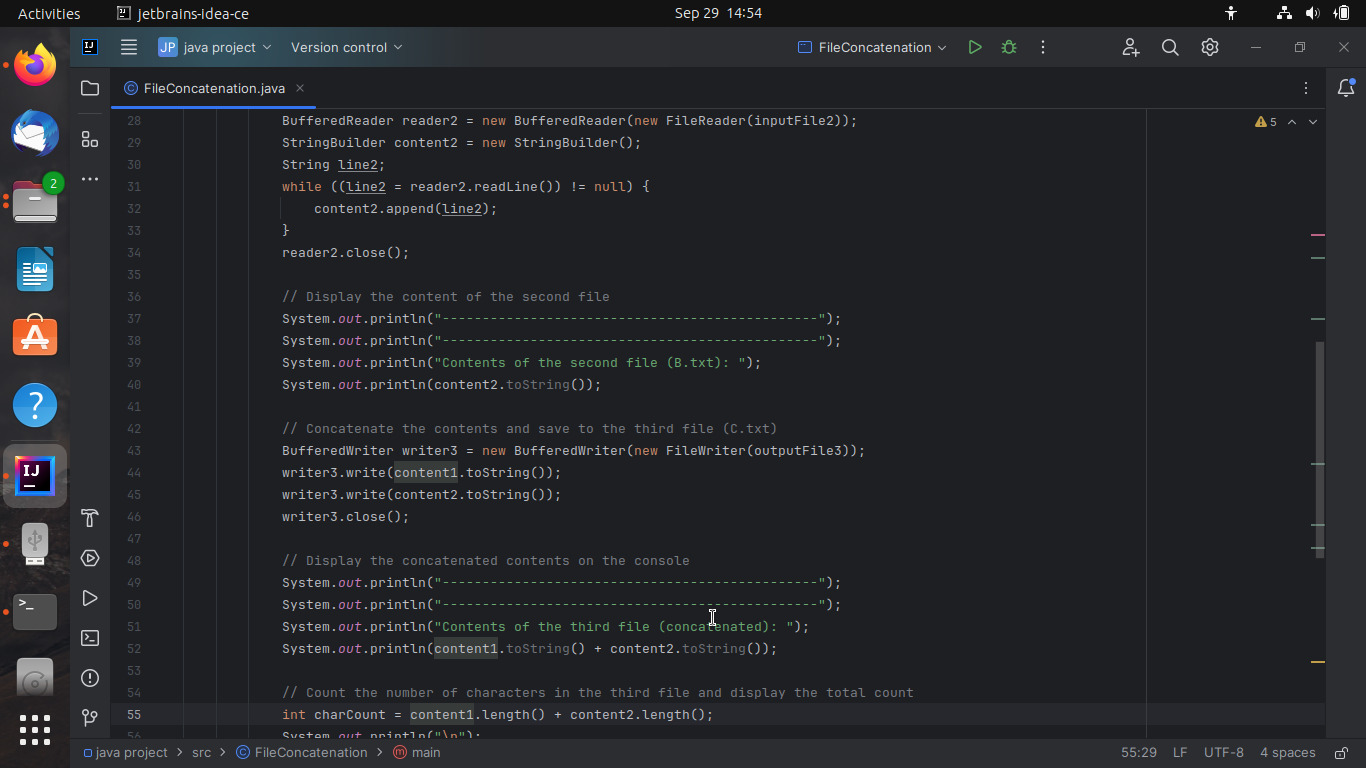
* Ensure you have the Java Development Kit (JDK) installed on your machine.
* Create a new project directory to keep your files organized.
* Create two text files within your project directory.
* Populate each file with the desired data for concatenation.
* Create a new Java class file in your project directory.
* Open the Java class file using a text editor or integrated development environment (IDE).
* Implement the logic to read data from the two input files and concatenate them.
* Save the concatenated data to a third file.
* Compile the Java class file using the Java compiler.
* Execute the compiled program to perform the file concatenation.
* Open the third file to ensure that the data from the input files has been correctly concatenated.

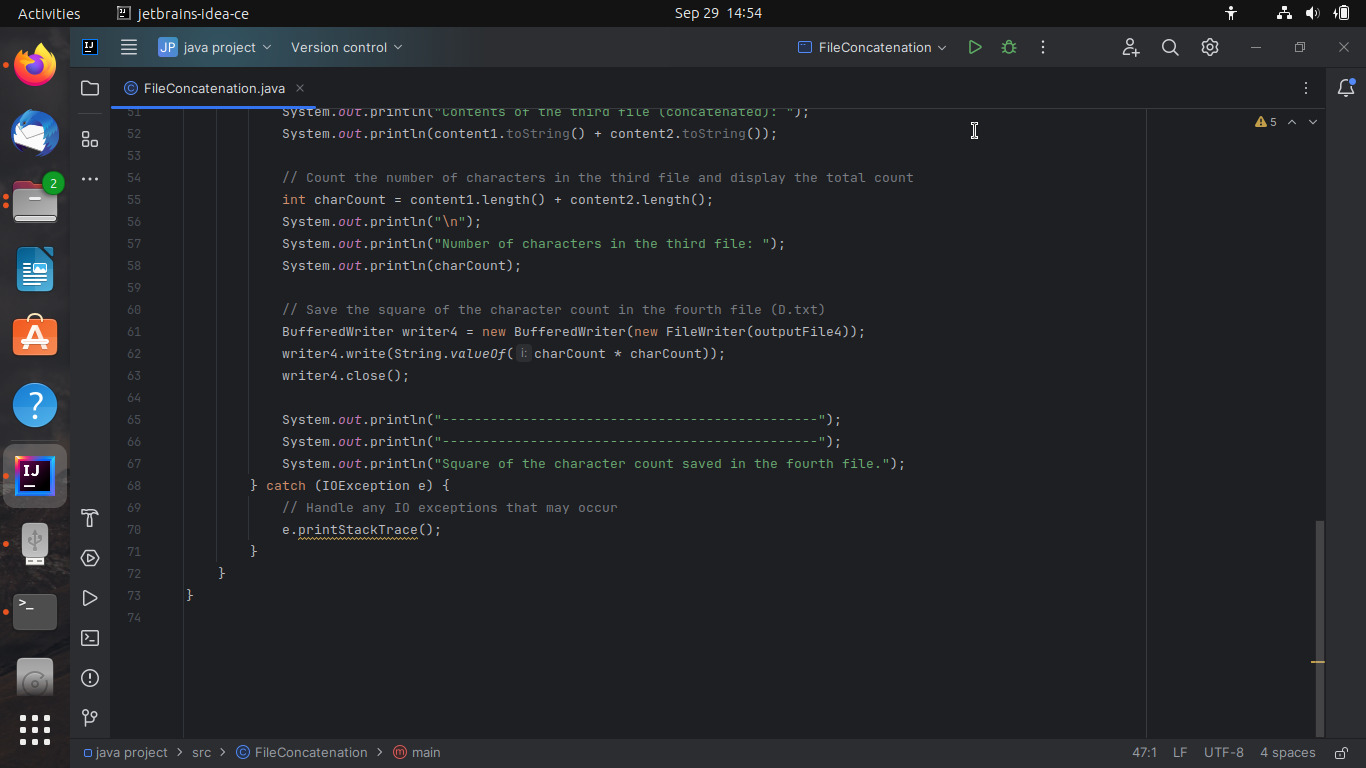
# Source Code

The complete java source code for the application.

**FileConcatenation.java:**







# Task Sheet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Description** | **Assigned to** | **Task Duration** | **Status** |
| Pseudo code | Create a clear, high-level outline of the program's logic to guide coding. | ALL TEAM MEMBERS | 1 hour | Completed |
| Actual coding and implementation | Set up the project, prepare input files, implement Java classes, and ensure character counting accuracy. | ANISIUBA MMESOMA BENEDICTA | 2 Hours 45 minutes | Completed |
| Flow chart | Create a visual flow chart to represent the program's execution flow. | OLISAEMEKA SOPHIA | 1 Hour | Completed |
| Data ﬂow diagram | Develop a data flow diagram illustrating data movement within the program. | AGU MARTHA | 1 Hour | Completed |
| Documentation | Write code comments, create user instructions, and prepare project report. | EGE OBETTA AMARACHI **and** KANU DANIEL CHIBUZOR | 5 Hours | Completed |
| Video Presentation | Prepare a video clip showcasing project functionality and features. | OCHINANWATA MAJESTY | 30 minutes | Completed |

# Checklist of Validation

|  |  |
| --- | --- |
| **Option** | **Validated** |
| Have all the modules been properly integrated and are completely functional? | Yes |
| Is exception handling mechanism implemented in all the modules? | Yes |
| Are all blocks of code, meaningfully commented? | Yes |
| Are all methods and classes properly named? | Yes |
| Are all the program codes working properly? | Yes |

# Submission Checklist

|  |  |  |
| --- | --- | --- |
| **Checkpoint** | **Status** | **Notes** |
| Has all modules been tested thoroughly? | Yes | Full program testing completed. |
| Does the program run adequately on the given software & hardware requirement? | Yes | The program was tested under the given requirement. |
| Does the Documentation give comprehensive details about the project? | Yes | Time was taken to make sure the Documentation accurately represented the project. |