Assembly Line Analysis Program

Marcus Timothy Ramos[1], Carlos Olivier R. Joyeux[2]
[1] Computer Engineering
[2] Industrial Engineering

marcus_ramos@dlsu.edu.ph, carlos_joyeux@dlsu.edu.ph

Abstract

This program is used to calculate the actual efficiency of an assembly line compared to its ideal efficiency. The first part is a sign in screen that is for organizing each assembly line. The usernames, passwords, and assembly line details will be stored in txt files (if we can make it work). After the user either signs in or creates a new account, they will be prompted to input the details of the assembly line (number of processes and ideal time for each process), these will be saved to the txt file (if possible). Next, the user is prompted to input the data for the actual current assembly line (amount of items, amount of failed items, and actual time for each process). The efficiency percent and failed/successful percent are then outputted for each process and as a whole along with a ranking of each process's efficiency, fail percent, and overall time taken.

- Prompts the user to sign in
- Collects the ideal processing time for each stage
- Collects the actual data about the assembly line
- Calculates what stages are inefficient compared to the ideal conditions

Introduction

This program is designed to aid students, especially engineering students to help them calculate the efficiency of an assembly line. By also having them to sign-in is a convenient way to add security to the program that we are about to create. It will ensure the program that the ones who will use it is an engineer who will compute for their assembly line. By just having the user to input their data for the assembly line, they can calculate and predict their progress and efficiency on their project in an instant.

Functionalities

This section describes the list of functionalities that you want to build as part of the application. Preferably, this list should be as granular as possible.

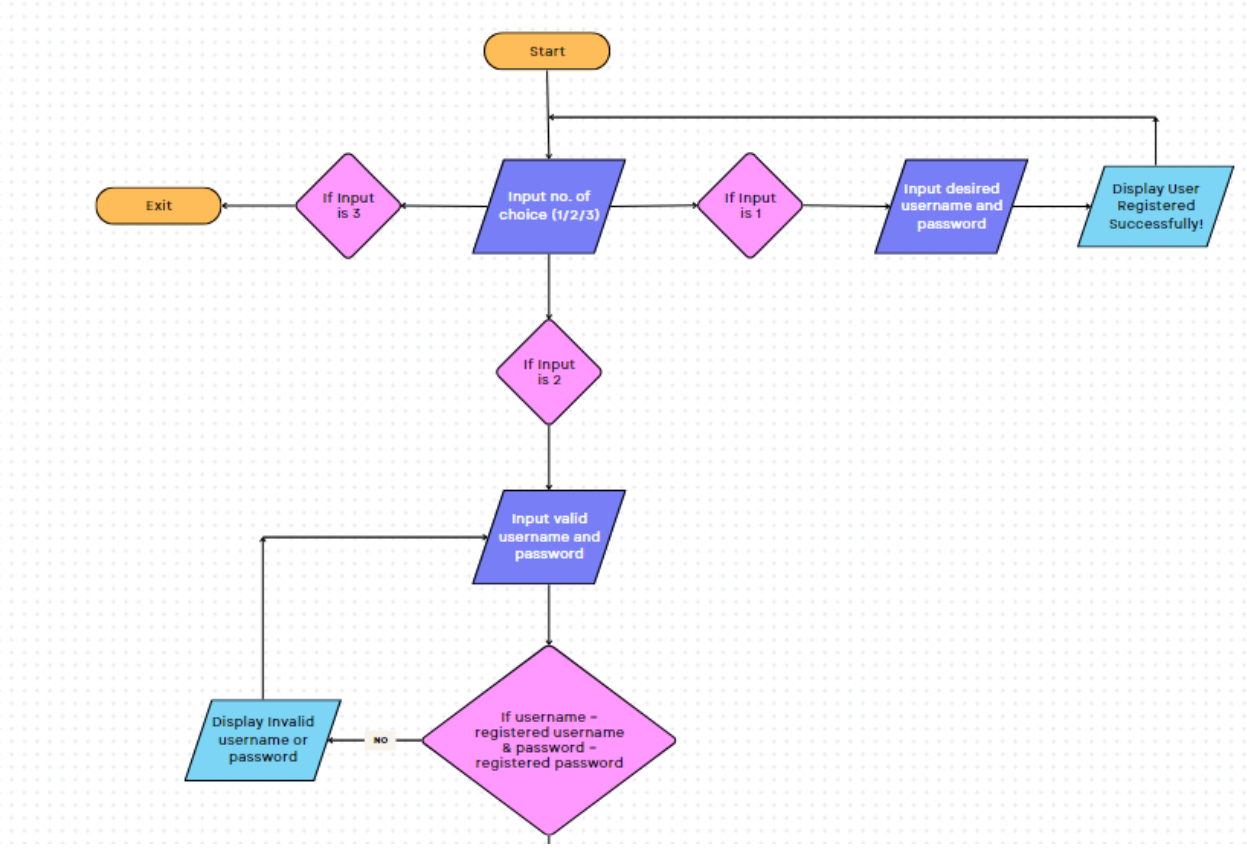
You can create these sets of functionality by formulating a table as shown below.

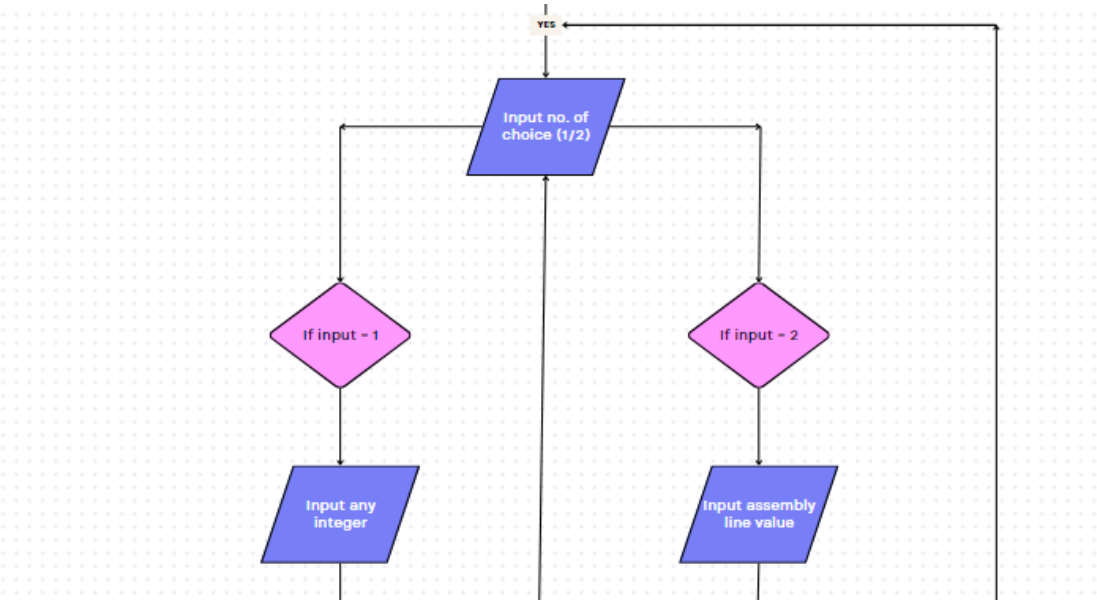
TABLE I. List of Functionalities

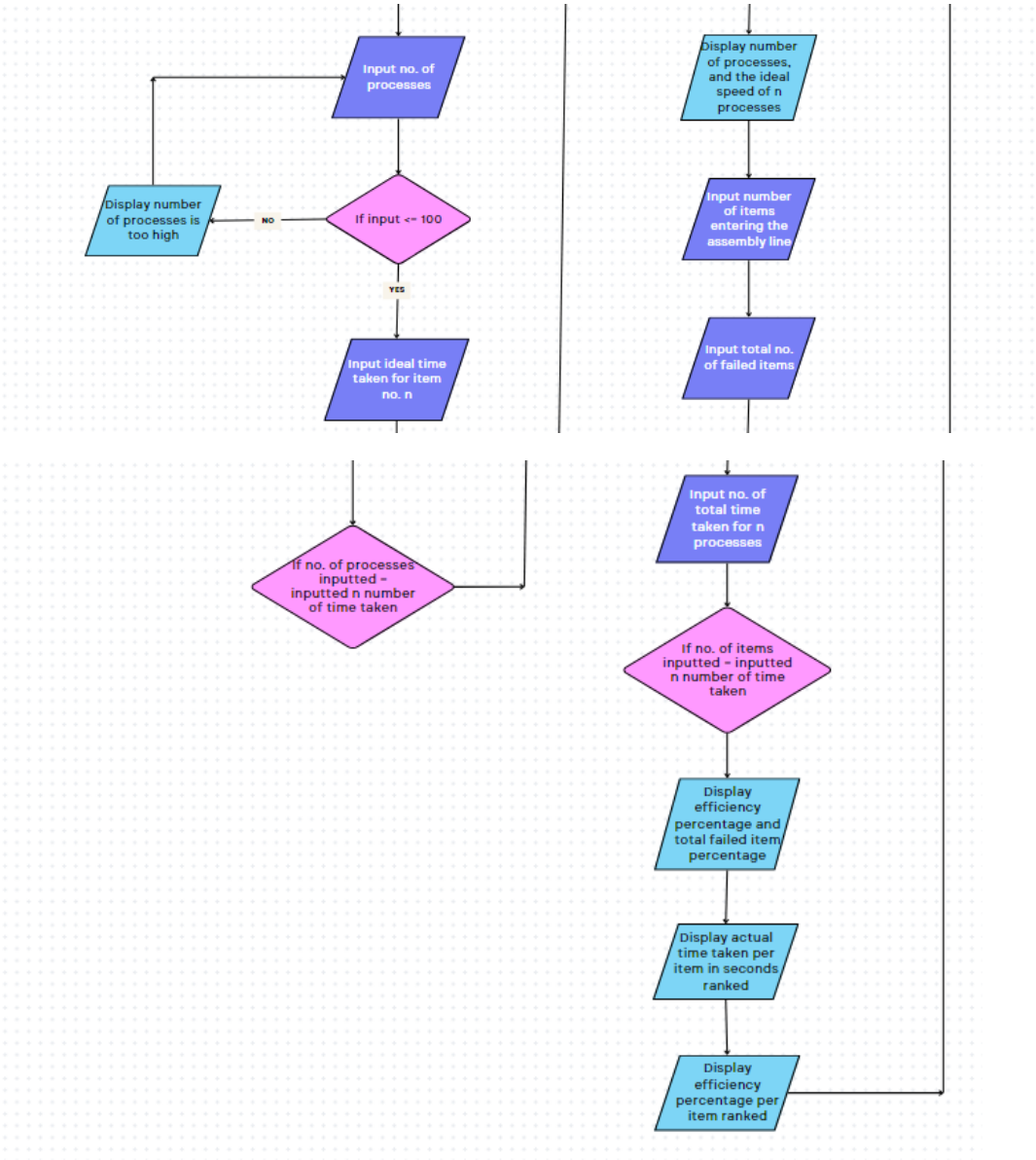
Persona	Description	Benefit
Production Planners/Schedulers, and Assembly Line Supervisors.	They need a place to keep the information about their production lines.	With the txt file system as a part of this program, they can store the ideal/average processing times and information about each individual stage.
Process Engineers and Quality Control Analysts	They need to compare the actual time for processes compared to the ideal time to identify bottlenecks and inefficiencies	The program would aggregate the data, calculate the efficiency, and separate it by each process.
Operations Managers and Cost Analysts.	They need to understand the time efficiency of production for cost analysis.	This program calculates the overall time for the process.
Senior Management and Investors	They need an evidence based overview on either how efficient their investment is or how efficient their management is.	This program provides useful metrics to base their judgments on.

System Development and Design

Flowchart







System Code/Pseudo-code

This section presents the system code or pseudo-code highlighting the modules covered or used. Include discussion and justification of the use of module/s.

Module 1: Introduction

Basic math calculations and standard input and output are used too frequently to state them all here but to explain simply, printf was used to display lines in the console, scanf and

getch(); were used to input variables into the code, and basic mathematical formulas were used to calculate the efficiency data.

Module 2: Conditional Statements

For the conditional statements, if/else is used constantly, ladderized if/else statements are used in the password input

```
while (i < MAX_LENGTH) {


        temp = getch();
        if(temp == 13){


            break;
        } else if (temp == 8 && i > 0) {
            i--;
            printf("\b \b");
        } else if (temp > 31 && temp < 127) {
            printf("*");
            password[i] = temp;
            i++;
        }
    }
```

And switch cases are used twice, once for choosing whether to login/register/exit

```
while (cont == 1) {
    printf("1. Register\n");
    printf("2. Login\n");
    printf("3. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &n);
```

```
        switch (n) {
            case 1:
                registerUser();
                break;
            case 2:
                        …
                        Break;
            case 3:
                exit(0);
            default:
                printf("Please Enter a Valid Input\n");
                getchar();
                getchar();
        }
    }
```

And switch is used to choose whether to alter/create a new ideal template or the test the actual data

```
while (cont == 1) {
        printf("\n1. Alter or Create a new Assembly line's Ideal Conditions");
        printf("\n2. Test Ideal Conditions of an Assembly Line against Actual Conditions\n");
        scanf("%d", &n);

        switch(n){
                case 1:
                        printf("\nWhat assembly line do you want to create or alter? (has to be an integer)\t");
                                scanf("%d", &assemblyLineNum);
                        enterSavedData(assemblyLineNum, username);
```

```
                    break;
            case 2:
                    …
                    Break;
            }
    }
```

Module 3: Functions

Functions were used to section the register, savedata, and datacompare functions.

```
    void registerUser() {
            …
    }
```

And,

```
    void enterSavedData(int assemblyLineNum, char username[MAX_LENGTH]) {
            …
    }
```

And lastly,

```
    void actualTest (int processes, int itemNum, int failedItemNum, float processtime[], float
    actualTime[]){
            …
    }
```

Module 4: Looping Statements

While loops are used for bugs in the switch statements like shown earlier.

And for loops are used to alter and sort arrays like I will show in the module 5 section.

Module 5: Arrays

Arrays are used most in the entersaveddata and actualtest function.

In the enterSavedData function an array is used to store the process time of each process

```
	for(i = 0; i < processes; i++) {
		processNum = i + 1;
		printf("\nIdeal Time taken (in seconds) per Item for process No.%d:\t",
		processNum);
		scanf("%f", &processtime[i]);
	}
```

And in the actualTest function, arrays are used for most of the code, but the most notable uses were for the bubble sort for the ranking of efficiency and time

```
for(i = 0; i < processes-1; i++){
		for(j = 0; j < processes-i-1; j++){
			if(actualTimePerItemTemp[j] > actualTimePerItemTemp[j+1]){
				temp = actualTimePerItemTemp[j];
				actualTimePerItemTemp[j] = actualTimePerItemTemp[j+1];
				actualTimePerItemTemp[j+1] = temp;
			}
		}
	}

	for(i = 0; i < processes; i++){
		for(j = 0; j < processes-i-1; j++){
			if(processEfficiencyTemp[j] < processEfficiencyTemp[j+1]){
				temp = processEfficiencyTemp[j];
				processEfficiencyTemp[j] = processEfficiencyTemp[j+1];
				processEfficiencyTemp[j+1] = temp;
			}
		}
	}
```

And the ranking parts

```
for(i = 0; i < processes; i++) {
        for(j = 0; j < processes; j++) {
                if (actualTimePerItemTemp[i] == actualTimePerItem[j]){
                        temp2 = j + 1;
                        actualTimePerItem[j] = -1;
                        break;
                }
        }
        printf("\nRank %d: Process No. %d at %f seconds per Item", i+1, temp2,
        actualTimePerItemTemp[i]);
}
```

And

```
for(i = 0; i < processes; i++) {
        for(j = 0; j < processes; j++) {
                if (processEfficiencyTemp[i] == processEfficiency[j]){
                        temp2 = j + 1;
                        processEfficiency[j] = -1;
                        break;

                }
        }
        printf("\nRank %d: Process No. %d at %f%%", i+1, temp2,
        processEfficiencyTemp[i] * 100);
}
```

Module 6: Strings

Strings are heavily used in the login/registration part of the program.

In the registration, strings are used to store the username and password

```
char username[MAX_LENGTH];

char password[MAX_LENGTH];
```

Next, in the login part strcmp is used to compare the inputted password to the sored one.

```
if (strcmp(password, storedPassword) == 0) {
    check = 1;  // Login successful
} else {
    check = 0;  // Login failed
}
```

Design of User Interface

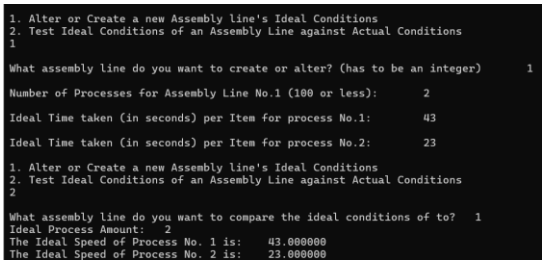
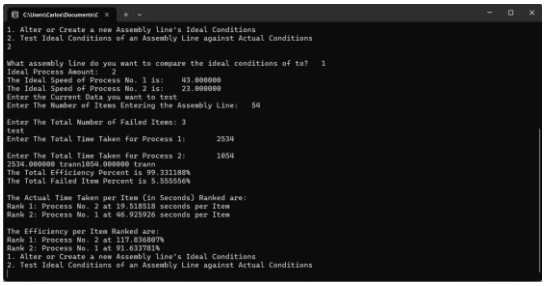
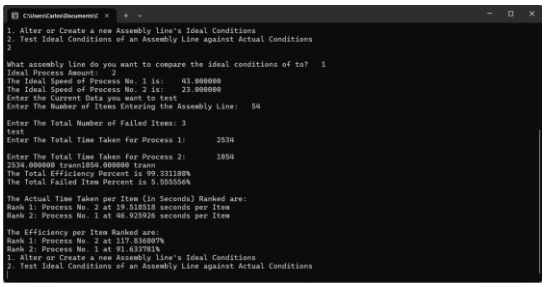
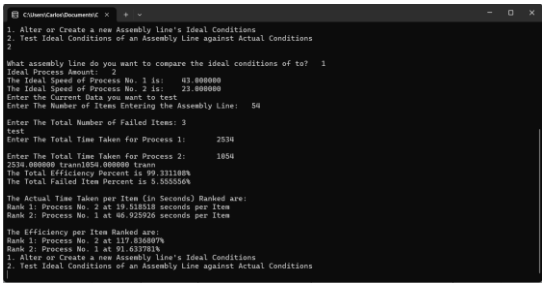
The user interface is a simple command prompt where the user shall input values , it can be numerical, alphabetical or a special character. Upon starting the program, the user shall follow instructions which are displayed on the program, otherwise the program shall prompt an error or invalid message, and it will loop back to the part where you left off. The user shall first register and login before using the program. Then, it will display a set of options where the user shall input numerical values to choose an option. Then, they will just follow the instructions displayed on the screen upon pressing enter. The program has a simple interface which is convenient and very easy to use.

System Output and Results

Functionality Displays

TABLE I. Functionality Output Display

Persona	Description	Output Display

Production Planners/Schedulers, and Assembly Line Supervisors.	They need a place to keep the information about their production lines.	
Process Engineers and Quality Control Analysts	They need to compare the actual time for processes compared to the ideal time to identify bottlenecks and inefficiencies	
Operations Managers and Cost Analysts.	They need to understand the time efficiency of production for cost analysis.	
Senior Management and Investors	They need an evidence based overview on either how efficient their investment is or how efficient their management is.	

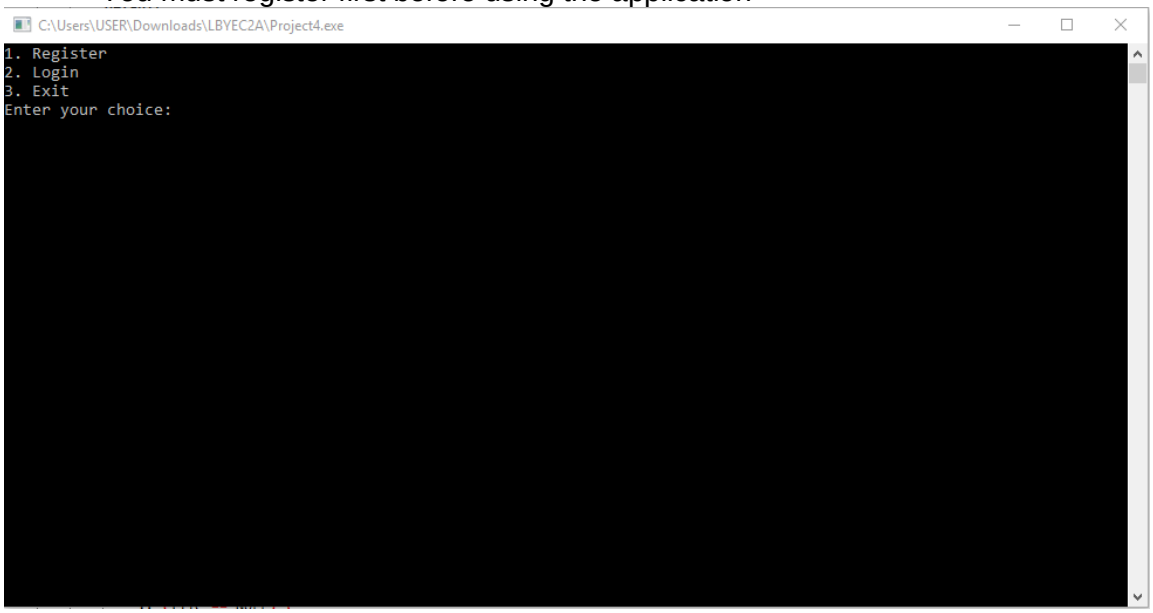
Step-by-step (Visualize) Technical User Guide

This section presents the step-by-step process of using the system using image screenshots from the running system with corresponding labels. Provide a discussion for each step.

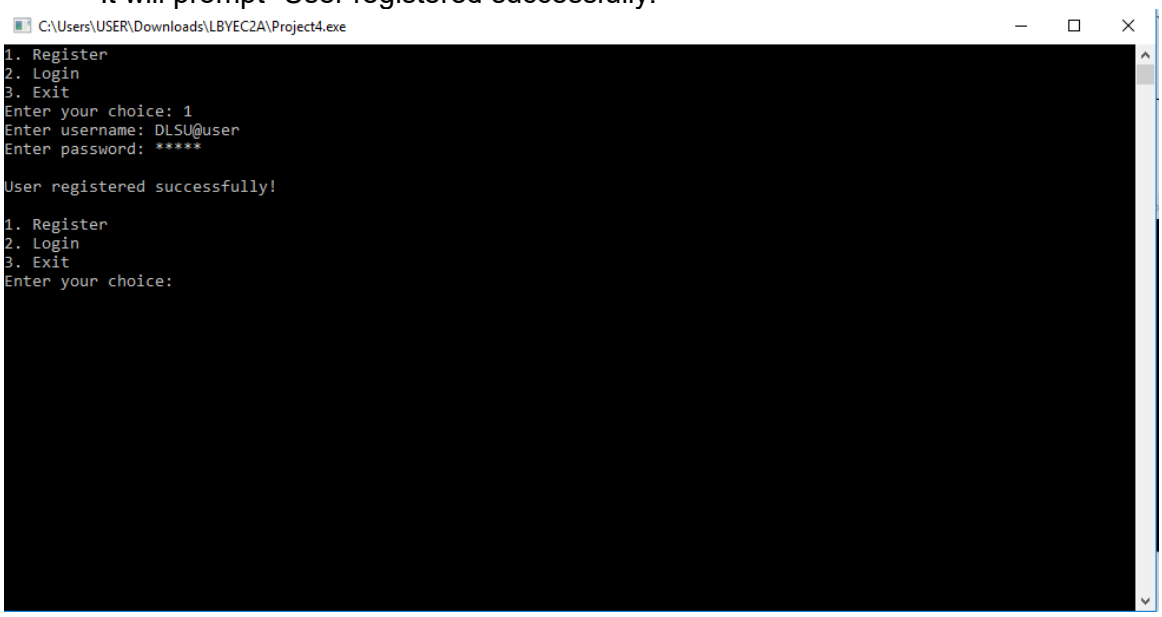
Sample:

Startup

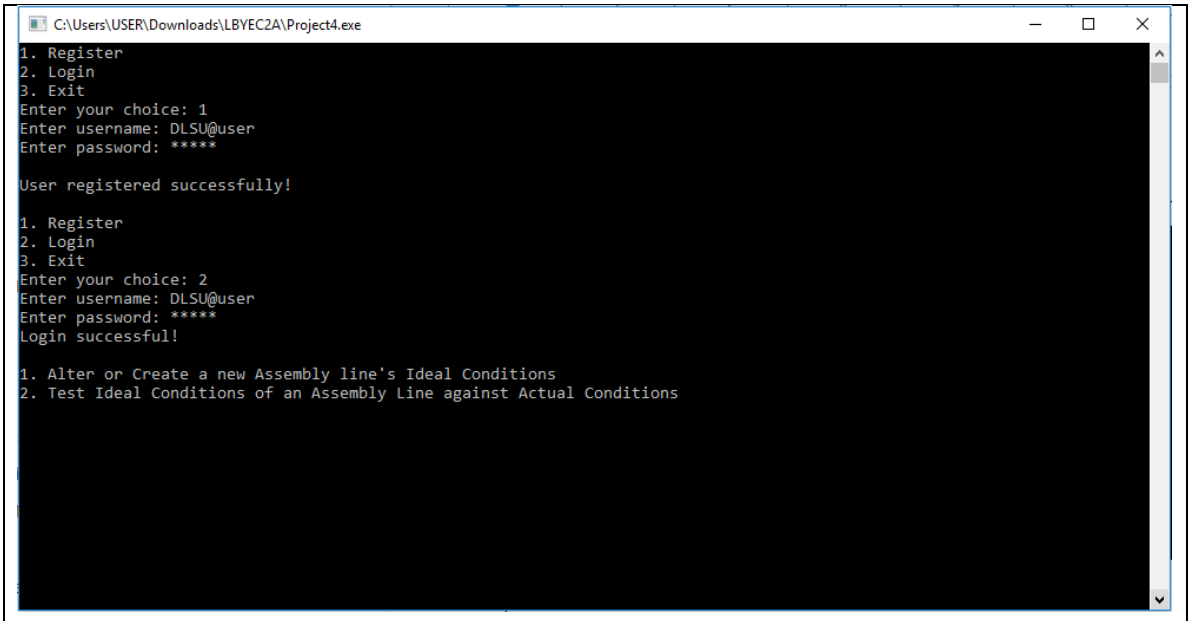
- Registration
- You must register first before using the application



- Select 1 to register into the application
- Enter your preferred username and password
- It will prompt “User registered successfully!”

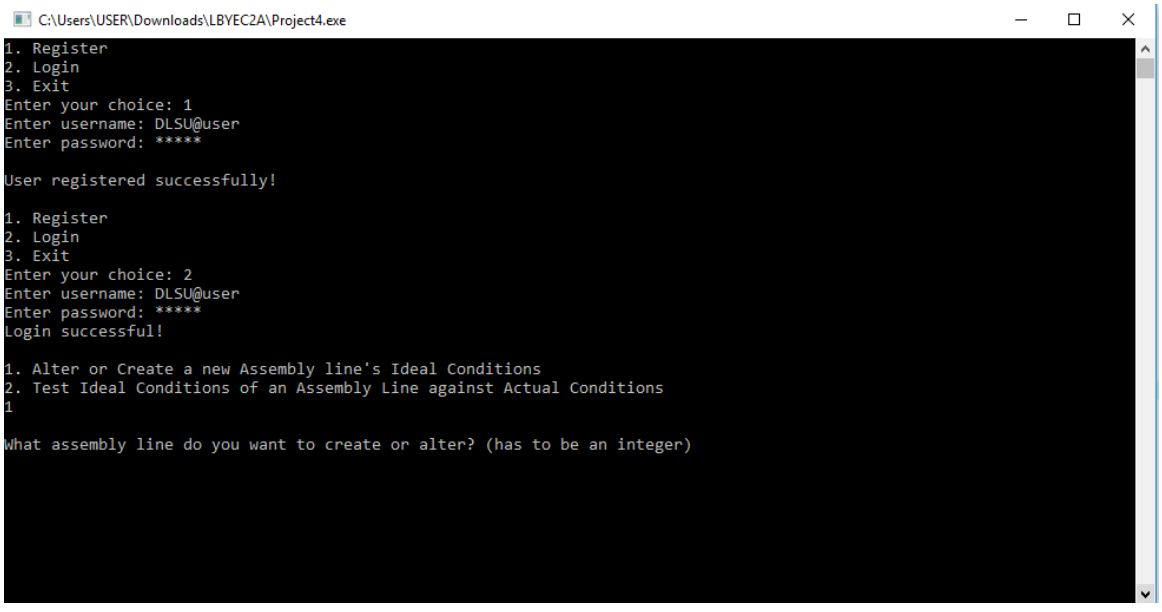


- Select 2 to login
- Login using your registered username and password
- It will prompt “Login successful!”

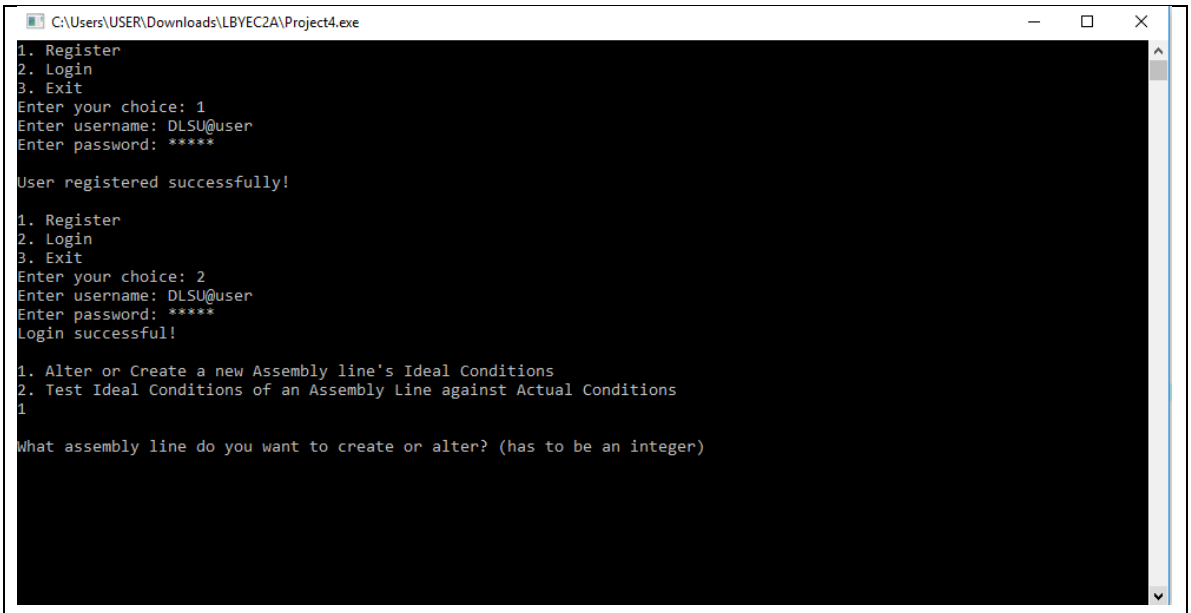


Main Features and Usage

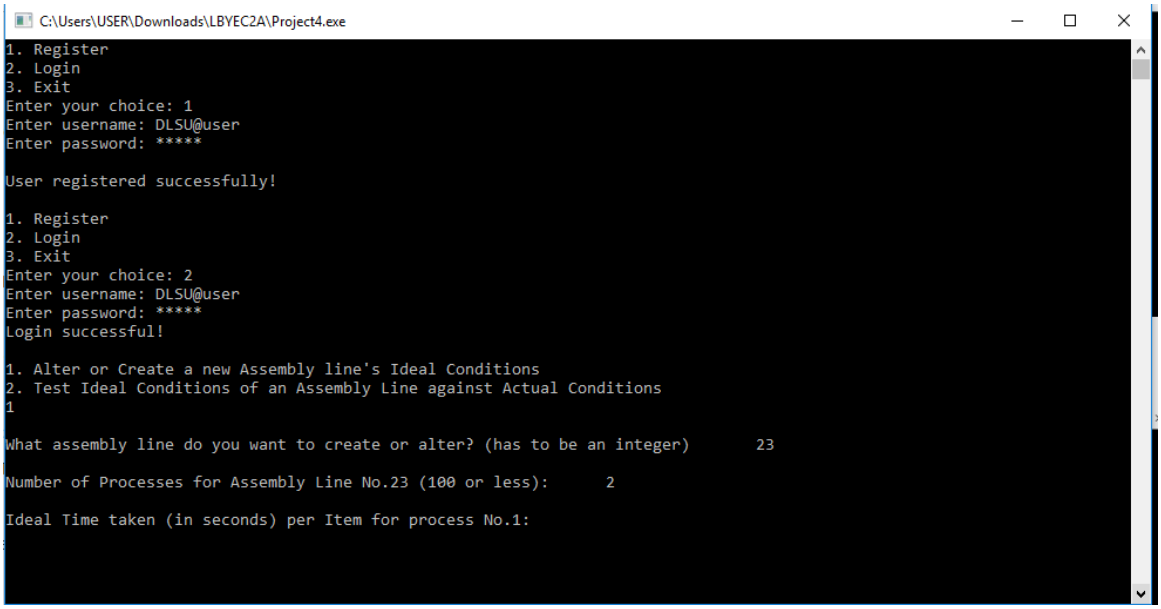
- **Using the assembly line program**
- **Feature 1**: Alter or Create a new Assembly line's Ideal Conditions
    - Select 1
    - It will ask you what assembly line do you want to create or alter



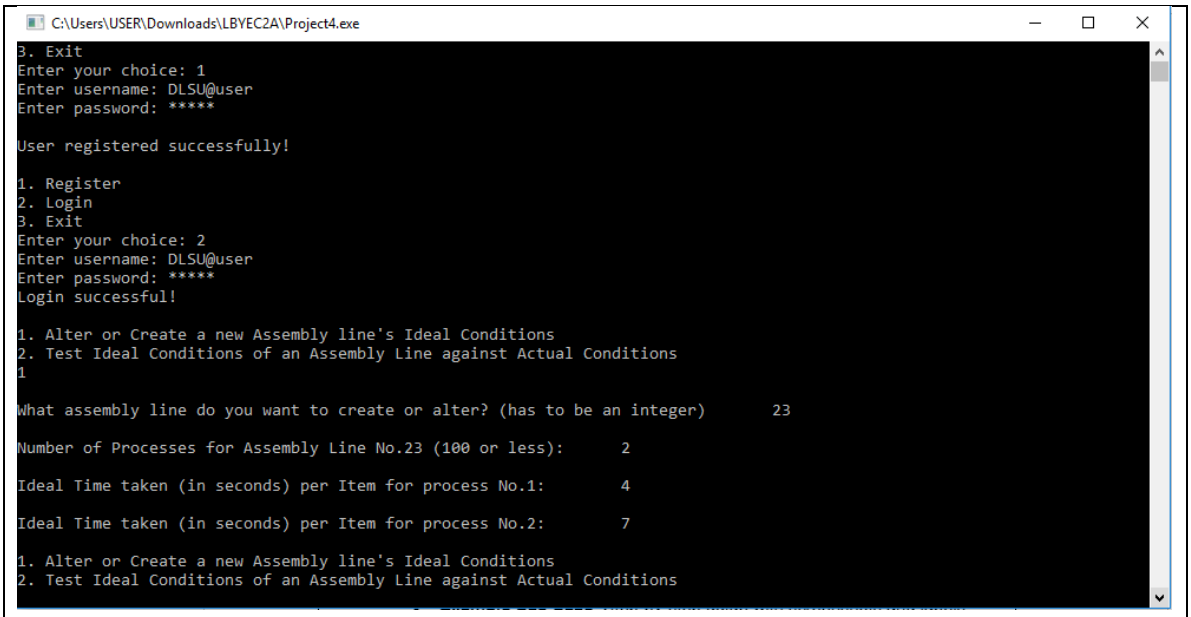
- Enter an integer
- It will ask you for the Number of Processes for Assembly Line No. (n)



- Enter the the Number of Processes for Assembly Line No. (n)
- It will ask you for the Ideal Time Taken for process no. (n)

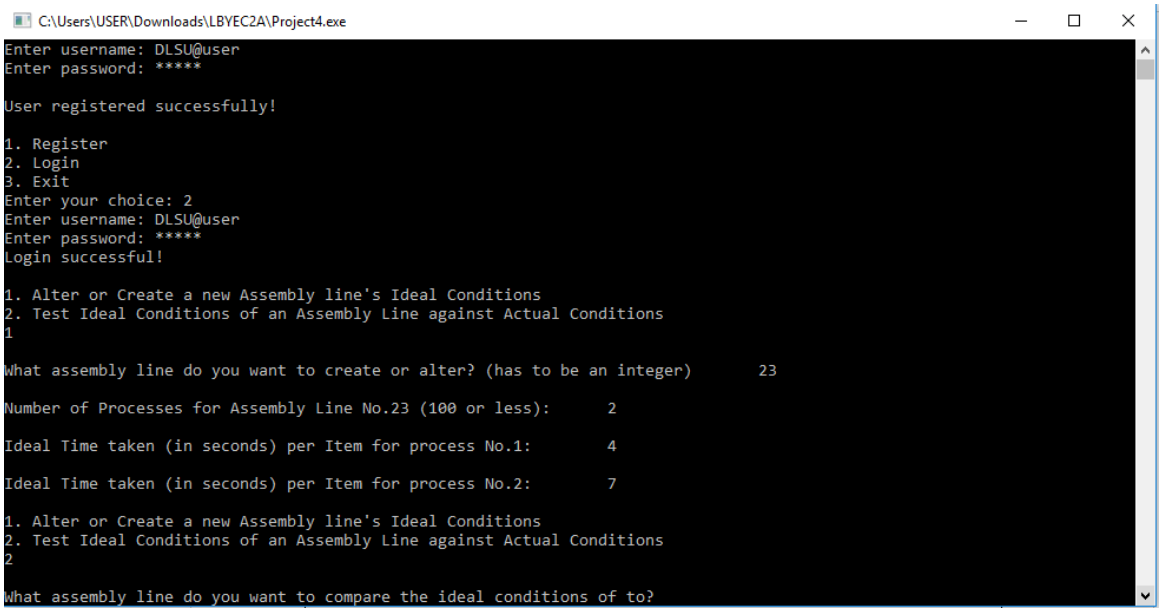


- Enter the ideal time taken for process no. (n)



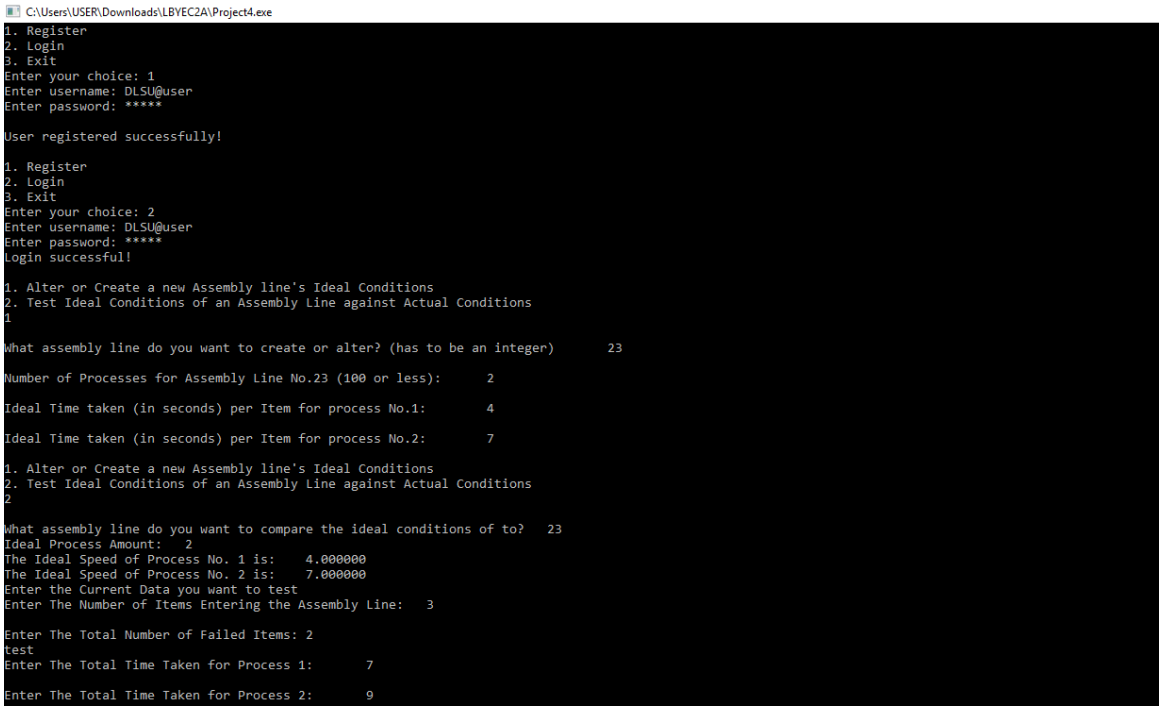
- The data is now stored

- **Feature 2**: Test Ideal Conditions of an Assembly Line against Actual Conditions
  - Select 2
  - It will ask you to enter the data of the assembly line that you want to compare the ideal conditions to



- Enter the numerical data of the assembly line that you stored

- It will now display the ideal process amount, and the ideal speed process of process no. (n)
- It will now ask you to enter the no. of items entering the assembly line
- It will also ask you to enter the no. of failed items
- Lastly, it will also ask you to enter the total time taken for process (n)



- It will now display the total efficiency percentages of each item
- It will also display the actual time taken of each item, which are ranked
- Lastly, it will display the efficiency rate of each item, which are also ranked

Conclusion and Future Recommendations

In summary, our final project for this term is an assembly line analysis program that takes inputs from the user and analyzes these inputs according to the functions in the assembly line. This program is very helpful for engineers to test the efficiency rate in the assembly line for their projects. This program would be helpful for them to predict the efficiency of their projects that are engineering-related, especially for students. It also has a feature where you can register before using this program, as an added security feature. Once registered, you can use the username and password you inputted to have access to the actual program. It is like a feature where it is exclusive for engineers. Our recommendations are that we could improve the program to look like an actual calculator or an organized program that takes input from the user neatly. We would also like to have the application as a separate program, which is what most applications have. Just by simply opening the application, then you can now access it. We could also ask for feedback on the future users of this program if they need more features and functions in the program.