**Datafication of PseudoCo.’s Biscuit Factory Machines**

Stephen Cornel,

Carl Kevin Gasal,

Jessica Marie Hernandez and

Akshar Devalkumar Patel

School of Engineering Technology and Applied Science, Centennial College

COMP120: Software Engineering Fundamentals

Syed Farhan Mazhar

December 8, 2022

Table of Contents

[Abstract 4](#_Toc121260121)

[Background 5](#_Toc121260122)

[Course Requirement 5](#_Toc121260123)

[Business Value 6](#_Toc121260124)

[Datafication of PseudoCo.’s Biscuit Factory Machines 6](#_Toc121260125)

[The Problem 7](#_Toc121260126)

[The Requirements 7](#_Toc121260127)

[The Solution 7](#_Toc121260128)

[Theory 7](#_Toc121260129)

[The Process Model 7](#_Toc121260130)

[Timeline 8](#_Toc121260131)

[Scope and Limitations 9](#_Toc121260132)

[User Interface and Input Method 9](#_Toc121260133)

[Visualization 9](#_Toc121260134)

[Accuracy 9](#_Toc121260135)

[Supported Devices 10](#_Toc121260136)

[Database Model 10](#_Toc121260137)

[Testing and Evaluation Strategy 11](#_Toc121260138)

[Development 12](#_Toc121260139)

[User Stories 12](#_Toc121260140)

[Issues Encountered, Solutions and the Pressing Timeline 13](#_Toc121260141)

[November 17: Display using Forms or CLI? 14](#_Toc121260142)

[November 18: Problem with saving data from user 16](#_Toc121260143)

[November 20: Displaying data from the database and calculation of values 17](#_Toc121260144)

[Results of Testing 19](#_Toc121260145)

[Black-box Test 19](#_Toc121260146)

[20](#_Toc121260147)

[Loop Test 21](#_Toc121260148)

[Conclusion and Recommendation 22](#_Toc121260149)

[References 23](#_Toc121260150)

[Appendix 25](#_Toc121260151)

[Meeting Minutes 25](#_Toc121260152)

[Flow Chart 28](#_Toc121260153)

[Jira Software 29](#_Toc121260154)

[Software Code 30](#_Toc121260155)

# Abstract

Since industries’ profits and losses are based on critical judgments, decision making must be logical and accurate, it must come in the form of “data”. Data is “factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation” (Merriam-Webster Dictionary, n.d.). According to Mejias and Couldry (2019), “the term ‘datafication’ implies that something is made into data”. To proceed with the goal of datafication, an issue must be raised, studied, and solved. By using the agile methodology, the group aimed to deliver the product owner’s requirements by using Python and SQLite3. Scrum framework was adopted to provide systematic approach to keeping the activities up to date and prioritization reasonable. The timeline is kept to 3 weeks that may accommodate 3 sprints. The challenge is to be able to come up with a working solution to the customer’s problem: PseudoCo.’s machines that cannot hit their daily quota because the factory does not have a reliable data management system. Although the group has limitations in terms of user interface, visualization, accuracy and supported devices, the results of the black-box test and loop test show that the requirements have been met. The members recommend that although the calculation of overall machine efficiency and defective ratio is not affected, the software must have data validation to enable user to input only the proper data types. Another recommendation is to make the query filter more flexible to accept any machine data instead of F01 to F03 only.

# Background

## Course Requirement

In partial fulfilment of COMP120: Software Engineering Fundamentals, this material serves as documentation of adopting the agile methodology in the product development, using SCRUM approach, into the process of creating a product, in the form of a software, that shall satisfy the assumed customer’s requirements within the stipulated time. The term project requires that the members focus on a latest technology. Latest technologies of 2023 include Computing Power, Smarter Devices, Artificial Intelligence and Machine Learning, Datafication, etc. (Duggal 2022). With the limited knowledge and programming capabilities of the members, the group opted for Datafication.

## Business Value

Lycett (2013) says that “From an industry perspective, BI is consequently seen as a fruitful foundation for innovation, competition and productivity”. The author adds that “Business Intelligence (BI) is broadening to encompass data infrastructure, applications, tools and best practices required for the effective capture, representation and delivery of data to inform decision making and action” (Lycett, 2013, para. 1). Since industries’ profits and losses are based on critical judgments, decision making must be logical and accurate, it must come in the form of “data”. Merriam-Webster dictionary defines data as “factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation” (Merriam-Webster Dictionary, n.d.). According to Mejias and Couldry (2019), “the term ‘datafication’ implies that something is made into data”. Furthermore, “to datafy a phenomenon is to put it in quantified form so that it can be tabulated and analyzed” (Mayer-Schönberger and Cukier 2013, p. 78). It is evident that companies will find that datafication is worth investing on; thus, developing a product that can make datafication possible will reap financial benefits. However, to proceed with the goal of datafication, an issue must be raised, studied, and solved.

## Datafication of PseudoCo.’s Biscuit Factory Machines

PseudoCo. is a biscuit company that built its first factory in Toronto just recently. For its initial production, it is using 3 machines. According to the planning department, 3 machines should be enough to deliver 100 boxes of chocolate biscuits and 50 boxes of plain biscuits per day. However, for the past 3 months, the factory has never hit their daily quota. They seek the help of a solutions company to understand their factory's problem by using their KPIs: machine efficiency and defective ratio.

### The Problem

The company does not know why they never hit their daily quota of 100 boxes of chocolate biscuits and 50 boxes of plain biscuits per day.

### The Requirements

Software should show machine efficiency and defective ratio.

### The Solution

Create a software that records each machine’s daily processing, compile it into a table and draw a bottleneck analysis on the factors that contribute to the productivity, machinery, or quality issue.

# Theory

## The Process Model

The traditional approach to software development is the ‘waterfall’ approach. However, because the PseudoCo. does not specify the complete requirements upfront, the business requirements might change during the project. Due to this vagueness, the members of the group are unable to set detailed breakdown of tasks and deliverables for the final product. In this regard, the timeline set will depend on how the whole team’s output progresses (referenced to Sherman, 2015). To rise to these challenges the group will use the agile methodology.

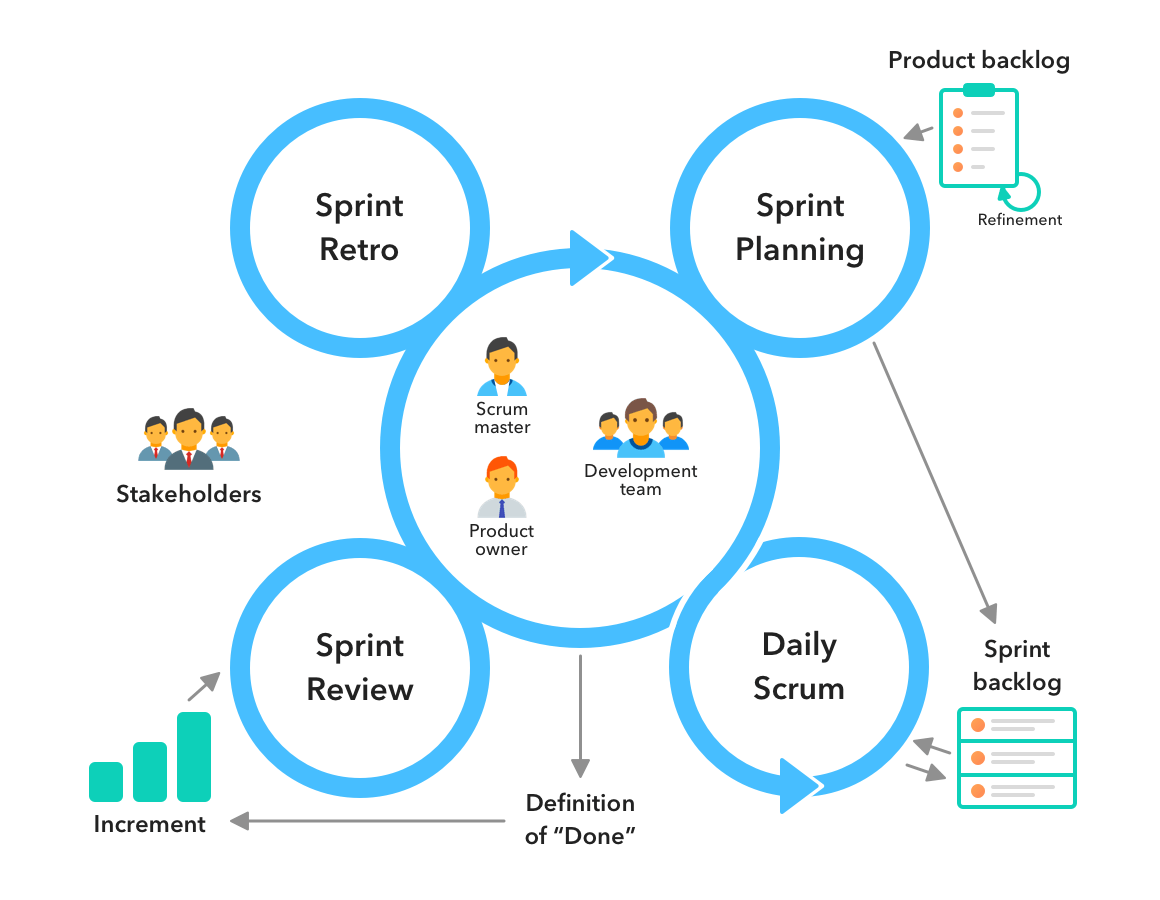
 “Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly” (Atlassian, n.d.). This methodology is what the group requires. Moreover, scrum, an agile framework, will provide structure for the team members to deliver value incrementally in a collaborative way (Scrum.org, n.d.).

Figure 1: Scrum Framework

## Timeline

The project is expected to last for 3 weeks, the sprints span from November 14 to December 2, 2022, including: sprint planning meetings (prior every sprint), scrum meetings (daily), and sprint reviews and retrospectives (every end of a sprint).

Whiteboard

Description automatically generated with medium confidence

Figure : Timeline of PseudoCo. Project

## Scope and Limitations

### User Interface and Input Method

Text

Description automatically generatedW3schools defines Command Line Interface (CLI) as “a command line program that accepts text input to execute operating system functions.” Additionally, “…with graphical user interfaces (GUI), most users never use command-line interfaces (CLI)”. With this, the users must bear in mind that no graphical user interface will be provided because the group will be offering a software programmed using Python and SQLite3 in CLI format.

Figure : Image of Command Line Interface

### Visualization

As the software will be packaged in executable file donning only black screen and some characters, the user must expect that graphs, charts and other forms of traditional visualizations will also not be present.

### Accuracy

Calculations are based on the understanding established within the group and have not been mathematically proven by experts on this field. The accuracy of data supplied by the software will be limited to the knowledge and capabilities of the group members.

### Supported Devices

The software can only run on Windows based desktop or laptop.

## Database Model

The database model is provided for better understanding of the relationship of different types of input (data) from the user in representing production facts in the line. The group will be utilizing SQLite3 for the database structure because Python will not accommodate database relationships and query (a request for data from a database) by itself, but it can import SQLite module to make this possible. “SQLite is a C library that provides a lightweight disk-based database that doesn’t require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It’s also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle” (Python, n.d.).

Figure : PseudoCo. Production Information Database

Graphical user interface, table

Description automatically generated

Figure : PseudoCo.'s Production Information Database

## Testing and Evaluation Strategy

According to IBM, “Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs, and improving performance.” Prior to deploying the product to the end-users, the group shall conduct the following testing strategies:

**Black-box testing** - Checking functions by emulating business scenarios, based on functional requirements.

**Loop testing** - is defined as a software testing type, that completely focuses on the validity of the loop constructs. It is one of the parts of Control Structure Testing (path testing, data validation testing, condition testing) (Guru99, n.d.)

On Figure 5, one of the loop tests will be to check if there will be no error if the user decides to exit ‘Display input/edit menu’ from B0 to Z0 by entering the characters 1 and 0, respectively.

Diagram

Description automatically generated

Figure : Part of PseudoCo. Production Information Database Flowchart

# Development

## User Stories

On November 15, 2022, the group established 4 main user stories; moreover, the members believe that these 4 epics encompasses the requirements of the product owner to guide the whole product development. These are:

1) As a user, I want the program to print what I input so that I can see all my entries.

2) As a user, I want to choose when to display the machine efficiency result so that I do not see data that I do not need.

3) As a user, I want to input machine information so that I don't have to record it on a paper.

4) As a user, I want the program to stay open to ask new batch of inputs so that I don't have to launch the program again.

Graphical user interface, application, Teams

Description automatically generated

Figure : Sprint 1 in JIRA Software

## Issues Encountered, Solutions and the Pressing Timeline

Initially, the group has decided on using Microsoft Visual Studio and C# to make the software. However, to respond to various difficulties that the members encountered while developing the product, the programming language and the software are unanimously agreed to be changed.

Graphical user interface, text, application, Teams

Description automatically generated

Figure : Deciding on the Programming Language

### November 17: Display using Forms or CLI?

The product owner is not strict with the visualization. The members of the group pondered on different types of user interface. The image on Figure 8 shows that if the group utilizes .NET forms on Microsoft Visual Studio, the user will have a more intuitive experience. However, the block is about the limitations of the members.

Graphical user interface

Description automatically generated

Figure : Model If Using C# and .NET Forms in Microsoft Visual Studio

Since the members are studying Python, this is the perfect opportunity to put those learnings to the test while further enhancing the members’ skills. This discussion is related to the next issue which is saving the data from user; moreover, it is recorded on the Jira project as shown on Figure 10.

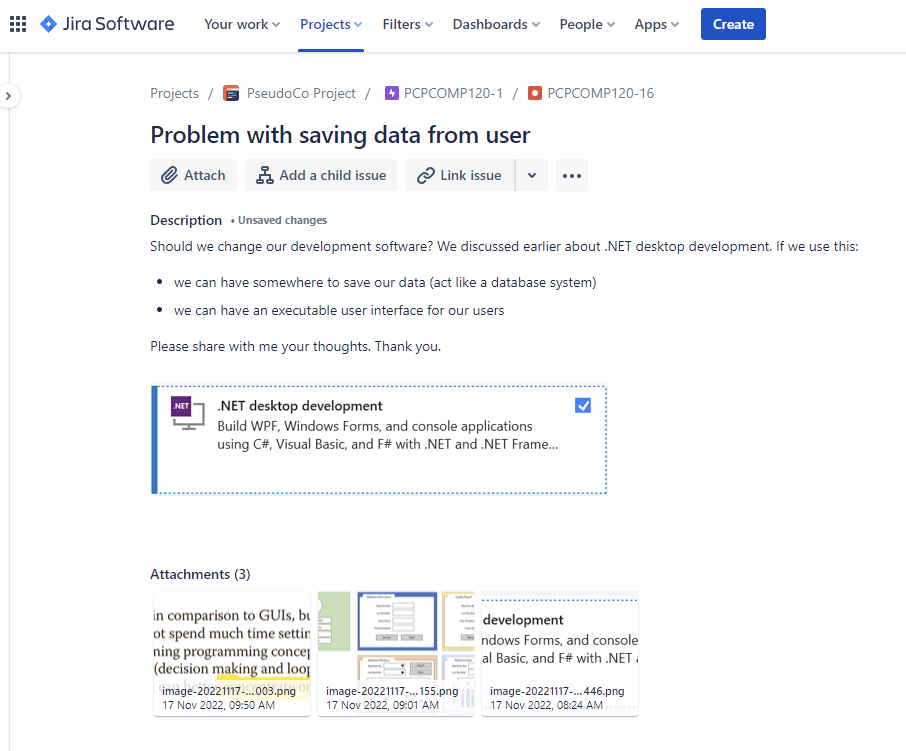


Figure : Jira Discussion on Finality of Software and Programming Language to Be Used

### November 18: Problem with saving data from user

The members are novice in the language of C#; therefore, the ability to save data from users is not that solid and dependable. Every time the user enters data, the program accepts it, but when it terminates, the data is cleared. Also, during the daily scrum meetings, the members realized that using C# without any means of saving the data externally, will only superficially display the assumed data but there is no capability to retrieve that data whenever the code calls for it. This is when the members searched for a different method of saving data.

Text

Description automatically generated

Figure : Displaying Saved Data on C#

There are 2 options: data tables using C# or SQLite using Python. Data tables are good in displaying tables inserted into the program, but the calculations are more complex than expected. Finally, using Python and importing SQLite3, the group can utilize query system as a means of database management.

### November 20: Displaying data from the database and calculation of values

Since the program has a database, it becomes easy to fetch filtered data directly from the created tables and assign them to variables. This enables the program to display the results of machine efficiency and defective ratio from the user input.

Text

Description automatically generated

Figure : Sample of Python Code (Machine Efficiency Calculation)

Text

Description automatically generated

Figure : Printing Calculation Results to Machine Data Report

Text

Description automatically generated

Figure : Machine Efficiency Report

In Figure 14, the program shows the machine efficiency report in a table because Python imports the tabulate module. This is to ensure that the printed data is neat and legible.

# Results of Testing

## Black-box Test

The black-box test includes inputting wrong data type and not inputting anything. The idea is that the program should display an error message and not crash. However, the program accepts wrong and no inputs.

# Text Description automatically generated

Figure : Accepting Wrong Data Type

Text

Description automatically generated

Figure : Accepting No Input

To check the repercussions of accepting wrong and no data to the database, the members tried checking machine efficiency before and after inputting wrong and no data.

Graphical user interface

Description automatically generated

Figure : Before and After Wrong and No Input

## Loop Test

To ensure that the program will not crash or suddenly stop working, the continuity of the program is checked using 22 tests formulated by the group. These tests are shown in Figure 18. The group finds that the software has no problem, and the paths are correctly followed. The loop tests are based on the flowchart created during planning.

Table

Description automatically generated with medium confidence

Figure : Results of Loop Test

# Conclusion and Recommendation

The software can meet the product owner requirements namely: printing the input, choosing when to display the data, inputting information and running the program continuously. The program may be used for inputting factory information of PseudoCo. factory’s machines from F01 to F03.

Still, the members recommend that although the calculation of overall machine efficiency and defective ratio is not affected, the software must have data validation to enable user to input only the proper data types. Another recommendation is to make the query filter more flexible to accept any machine data instead of F01 to F03 only.

# References

Duggal, N. (2022, December 5). Top 18 New Technology Trends for 2023. *Simpli Learn*. <https://www.simplilearn.com/top-technology-trends-and-jobs-article#3_datafication>

Hamilton, T. (2022, November 5). What is Loop Testing? Methodology, Example. *Guru 99.* <https://www.guru99.com/loop-testing.html>

Infinity. (n.d.). Scrum Methodology: An Agile Way of Managing Projects. In *Star Infinity*. Retrieved November 29, 2022, from <https://startinfinity.com/project-management-methodologies/scrum>

Lycett, M. (2013). 'Datafication': making sense of (big) data in a complex world.*European Journal of Information Systems,*22(4), 381-386. <https://doi.org/10.1057/ejis.2013.10>

Mejias, U. A. & Couldry, N. (2019). Datafication. *Internet Policy Review*, 8(4). <https://doi.org/10.14763/2019.4.1428>

Merriam-Webster, Incorporated. (n.d.). Data. In *Merriam-Webster Dictionary*. Retrieved November 25, 2022, from <https://www.merriam-webster.com/dictionary/data>

Python Software Foundation. (n.d.). sqlite3 — DB-API 2.0 interface for SQLite databases. In *Python.org.* Retrieved November 29, 2022, from <https://docs.python.org/3/library/sqlite3.html>

Sherman, R. (2015). Project Management. *Business Intelligence Guidebook*, 2015, Pages 449-492. <https://doi.org/10.1016/B978-0-12-411461-6.00018-6>

W3Schools. (n.d.). What is Command Line Interface (CLI)? In *W3Schools*. Retrieved November 29, 2022, from <https://www.w3schools.com/whatis/whatis_cli.asp>

# Appendix

## Meeting Minutes

|  |  |  |
| --- | --- | --- |
| **Date** | **Type** | **Details** |
| 14-Nov-22 | 1st Sprint Planning | Attendees: Carl, Stephen, Jessica, Akshar  Role Assignment:   * Scrum master: Carl * Product owner: Stephen, Akshar * Developer: Jessica   Communication tool: Microsoft Teams  Project collaboration software: Jira  Programming Language: C#  Programming Software: Microsoft Visual Studio  Target to finish first sprint: November 19, 2022  First sprint contents:   * Decide on programming language to use * Decide which software development tool to use * Identify inputs from Pseudo Co. machines * Create user interface for data processing * Calculate machine efficiency * Decided order of priority per epic |
| 15-Nov-22 | Daily Scrum Meeting | Attendees: Carl, Stephen, Jessica, Akshar  [Last Meeting Accomplishment]  [Obstacles]  [Next Meeting Target] Focus on generated user stories:  1) As a user, I want to input machine information so that I don’t have to record it on paper.  2) As a user, I want the program to print what I input so that I can see all my entries.  3) As a user, I want the program to stay open to ask new batch of inputs so that I don’t have to launch the program again.  4) As a user, I want to choose when to display the machine efficiency result so that I do not see data that I do not need. |
| 16-Nov-22 | Daily Scrum Meeting | Attendees: Carl, Stephen, Jessica, Akshar  [Last Meeting Accomplishment] Input data from user  [Obstacles] Printing entries from user in a table, re-entering details from user prompt not showing  [Next Meeting Target] Use loop, case and main menu screen prior to user input |
| 17-Nov-22 | Daily Scrum Meeting | Attendees: Carl, Jessica  [Last Meeting Accomplishment] Using Jira for code sharing  [Obstacles] Figure out how to loop printing inputs by user  [Next Meeting Target] Continue looping inputs, calculate machine efficiency considering cumulative averages |
| 20-Nov-22 | 2nd Sprint Planning | Attendees: Carl, Stephen, Jessica, Akshar   * Marked 1st Sprint as complete with 5 open activities moved to 2nd sprint * Started 2nd Sprint * Discussed how to access pull up request for modification of codes on Jira and Bitbucket * Discussed existing problems with current open activities |
| 22-Nov-22 | Daily Scrum Meeting | Attendees: Carl, Stephen, Jessica, Akshar  [Last Meeting Accomplishment] Printing all input from user  [Obstacles] Only prints when inputting details, computation defective ratio using DataTable (C#, MS Visual Studio)  [Next Meeting Target] Separate input method from display method for machine efficiency, provide existing data for machine information and defect |
| 23-Nov-22 | Daily Scrum Meeting | Attendees: Carl, Stephen, Jessica, Akshar  [Last Meeting Accomplishment] Separate input and display methods and provide (fake) existing data for analysis and computation for machine efficiency, used Python and SQLite3 due to database system requirement  [Obstacles]  [Next Meeting Target] Use Python and SQLite3 for input, display, and calculation of defective ratio |
| 25-Nov-22 | Daily Scrum Meeting | Attendees: Carl, Stephen, Jessica, Akshar  [Last Meeting Accomplishment] Input and display quality reports  [Obstacles] Insufficient information to provide analysis on productivity, machinery, and quality root causes  [Next Meeting Target] Calculate and display defective ratio and machine efficiency factors distribution |
| 27-Nov-22 | 3rd Sprint Planning | Attendees: Carl, Stephen, Jessica, Akshar   * Marked 2nd Sprint as complete with 3 open activities moved to 3rd sprint * Started 3rd Sprint * Discussed how executable version of the program and changed user interface * Discussed existing problems with current open activities |
| 29-Nov-22 | Daily Scrum Meeting | Attendees: Carl, Stephen, Jessica, Akshar  [Last Meeting Accomplishment] Program finished for testing made executable with pre-installed user data  [Obstacles] What type of testing to use and who will do the tests  [Next Meeting Target] Results of testing: Jessica (loop test), Stephen (black box test), and Carl (independent group test) |
| 2-Dec-22 | Sprint Review and Retrospect | Attendees: Carl, Stephen, Jessica, Akshar   * Test results: no data validation so program accepts any data type and prints results to View Performance menu * Results of calculation for machine efficiency and defective ratio are not affected by wrong data types. * The time to make the project is short but the team managed to develop a working product. |

Note: Meetings are recorded in Microsoft Teams

Link:<https://teams.microsoft.com/l/team/19%3aX42unBrmxsf9bQk3jVGsxJI537vUbR4jp4j2WRmrCIw1%40thread.tacv2/conversations?groupId=f7644ad7-707a-481a-aec0-541031e3c18a&tenantId=316aa4d3-f4ee-48f0-9ce7-cef2e1e2a331>

## Flow Chart

Graphical user interface, diagram

Description automatically generated

## Jira Software

Graphical user interface, application, Teams

Description automatically generated

Link to Jira Project:

<https://circarl.atlassian.net/jira/software/projects/PCPCOMP120/boards/2/roadmap?shared=&atlOrigin=eyJpIjoiMjhjZWFlYzhhNTc5NDExOGI1MjA4MTc0MzM3N2MzMzciLCJwIjoiaiJ9>

## Software Code

import sqlite3

# Global Declarations

divider = "-"

menu\_0 = "0) Exit"

#Create table

conn = sqlite3.connect('factory.db')

c = conn.cursor()

# Methods for Window Sequence

def under\_construction():

    print("\nSystem Response:")

    print("We're sorry, this feature is under construction.")

    print("You will be navigated back to the main menu.")

    print("Thank you for understanding.")

    main\_menu()

def get\_main\_menu\_response():

    response = input("Enter 1 or 3 to continue or 0 to exit: ")

    if response == '1':

        input\_data()

    if response == '2':

        view\_perf()

    if response == '3':

        lot\_data\_db()

def get\_input\_response():

    response = input("Enter 1 to 4 to continue or 0 to exit: ")

    if response == '1':

        lot\_info()

    if response == '2':

        machine\_data()

    if response == '3':

        quality\_data()

    if response == '4':

        main\_menu()

def get\_view\_response():

    response = input("Enter 1 to 5 to continue or 0 to exit: ")

    if response == '1':

        machine\_data\_db()

    if response == '2':

        quality\_data\_db()

    if response == '3':

        machine\_efficiency()

        main\_menu()

    if response == '4':

        defective\_ratio()

        main\_menu()

    if response == '5':

        main\_menu()

def main\_menu():

    header = "Welcome to PseudoCo."

    length = len(header)+15

    subheader = "Production Information Database"

    menu\_1 = "1) Input/Edit Data"

    menu\_2 = "2) View Performance"

    menu\_3 = "3) View Registered Lots"

    print("\n" + header.center(length," "))

    print(subheader.center(length," "))

    print(divider.center(length,"-"))

    print("\t"+ menu\_1.ljust(length, " "))

    print("\t"+ menu\_2.ljust(length, " "))

    print("\t"+ menu\_3.ljust(length, " "))

    print("\t"+ menu\_0.ljust(length, " ")+"\n")

    get\_main\_menu\_response()

def input\_data():

    header = "Input/Edit Data"

    length = len(header)+15

    menu\_1 = "1) Lot Information"

    menu\_2 = "2) Machine Data"

    menu\_3 = "3) Quality Data"

    menu\_4 = "4) Go back to Main Menu"

    print("\n"+header.ljust(length," "))

    print(divider.center(length,"-"))

    print(menu\_1.ljust(length, " "))

    print(menu\_2.ljust(length, " "))

    print(menu\_3.ljust(length, " "))

    print(menu\_4.ljust(length, " "))

    print(menu\_0.ljust(length, " ")+"\n")

    get\_input\_response()

def view\_perf():

    header = "View Performance"

    length = len(header)+15

    menu\_1 = "1) Machine Data Reports"

    menu\_2 = "2) Quality Data Reports"

    menu\_3 = "3) Machine Efficiency"

    menu\_4 = "4) Defective Ratio"

    menu\_5 = "5) Go back to Main Menu"

    print("\n"+header.ljust(length," "))

    print(divider.center(length,"-"))

    print(menu\_1.ljust(length, " "))

    print(menu\_2.ljust(length, " "))

    print(menu\_3.ljust(length, " "))

    print(menu\_4.ljust(length, " "))

    print(menu\_5.ljust(length, " "))

    print(menu\_0.ljust(length, " ")+"\n")

    get\_view\_response()

def lot\_info():

    header = "Lot Information"

    length = len(header)+15

    print("\n"+header.ljust(length," "))

    print(divider.center(length,"-"))

    response = input("Enter 1: enter info, 2: edit, 3: go back to main menu, 0: exit ")

    if response == '3':

        main\_menu()

    if response == '2':

        under\_construction()

    if response == '1':

        lot\_num = input("\nLot Number: ")

        biscuit\_type = input("Type: ")

        lot\_qty = input("Lot Qty: ")

        mc\_total\_time = input("Machine Processing Time: ")

        plan\_date = input("Plan Date (yyyy/mm/dd): ")

        #save information on sqlite

        lot\_info = [(lot\_num,biscuit\_type,lot\_qty,mc\_total\_time,plan\_date)]

        c.executemany("INSERT INTO lot\_information VALUES (?,?,?,?,?)",lot\_info)

        conn.commit()

        print("\nYour entry has been recorded!")

        main\_menu()

def machine\_data():

    header = "Machine Data"

    length = len(header)+15

    print("\n"+header.ljust(length," "))

    print(divider.center(length,"-"))

    response = input("Enter 1: enter info, 2: edit, 3: go back to main menu, 0: exit ")

    if response == '3':

        main\_menu()

    if response == '2':

            under\_construction()

            #will have to open sqlite and filter data according to lot number

    if response == '1':

        machine\_num = input("\nMachine Number: ")

        lot\_num = input("Lot Number: ")

        downtime\_factor = input("Downtime Factor: ")

        downtime = input("Downtime: ")

        runtime = input("Runtime: ")

        process\_date = input("Process Date (yyyy/mm/dd): ")

        #save information on sqlite

        machine\_data = [(machine\_num,lot\_num,downtime\_factor,downtime,runtime,process\_date)]

        c.executemany("INSERT INTO machine\_data VALUES (?,?,?,?,?,?)",machine\_data)

        conn.commit()

        print("\nYour entry has been recorded!")

        main\_menu()

def quality\_data():

    header = "Quality Data"

    length = len(header)+15

    print("\n"+header.ljust(length," "))

    print(divider.center(length,"-"))

    response = input("Enter 1: enter info, 2: edit, 3: go back to main menu, 0: exit ")

    if response == '3':

        main\_menu()

    if response == '2':

            under\_construction()

            #will have to open sqlite and filter data according to lot number

    if response == '1':

        lot\_num = input("\nLot Number: ")

        defect\_mode = input("Defect Mode: ")

        defect\_qty = input("Defect Qty: ")

        process\_date = input("Process Date (yyyy/mm/dd): ")

        #save information on sqlite

        quality\_info = [(lot\_num,defect\_mode,defect\_qty,process\_date)]

        c.executemany("INSERT INTO quality\_data VALUES (?,?,?,?)",quality\_info)

        conn.commit()

        print("\nYour entry has been recorded!")

        main\_menu()

# Methods for database

def delete\_tables():

    c.execute('''DROP TABLE lot\_information''')

    c.execute('''DROP TABLE machine\_data''')

    c.execute('''DROP TABLE quality\_data''')

    print("Tables have been deleted.")

def create\_tables():

    c.execute('''CREATE TABLE IF NOT EXISTS lot\_information

             (lot\_num text,

             biscuit\_type text,

             total\_qty integer,

             mc\_total\_time integer,

             plan\_date text)

''')

    c.execute('''CREATE TABLE IF NOT EXISTS machine\_data

             (machine\_num text,

             lot\_num text,

             downtime\_factor text,

             downtime integer,

             runtime integer,

             process\_date text)

''')

    c.execute('''CREATE TABLE IF NOT EXISTS quality\_data

             (lot\_num text,

             defect\_mode text,

             defect\_qty integer,

             process\_date text)

''')

    print("Success! Tables have been created.")

# Methods for displaying database contents

def machine\_data\_db():

    length = 80

    header = " MACHINE DATA REPORT "

    footer = " END OF REPORT "

    print("\n"+header.center(length, '\*')+"\n")

    c.execute("SELECT rowid, \* FROM machine\_data")

    items = c.fetchall()

    from tabulate import tabulate

    print(tabulate(items, headers=["No.","Machine","Lot No.","Factor", "Downtime", "Date"]))

    print("\n"+footer.center(length, '\*'))

    main\_menu()

def quality\_data\_db():

    length = 80

    header = " QUALITY DATA REPORT "

    footer = " END OF REPORT "

    print("\n"+header.center(length, '\*')+"\n")

    c.execute("SELECT rowid, \* FROM quality\_data")

    items = c.fetchall()

    from tabulate import tabulate

    print(tabulate(items, headers=["No.","Lot No.","Defect Mode", "Defect Qty", "Date"]))

    print("\n"+footer.center(length, '\*'))

    main\_menu()

def lot\_data\_db():

    length = 80

    header = " REGISTERED LOTS REPORT "

    footer = " END OF REPORT "

    print("\n"+header.center(length, '\*')+"\n")

    c.execute("SELECT rowid, \* FROM lot\_information")

    items = c.fetchall()

    from tabulate import tabulate

    print(tabulate(items, headers=["No.","Lot Number","Biscuit Type","Lot Qty", "Processing Time", "Date"]))

    print("\n"+footer.center(length, '\*'))

    main\_menu()

# Methods for calculations

def machine\_efficiency():

    header = " MACHINE EFFICIENCY REPORT "

    footer = " END OF REPORT "

    print("\n"+header.center(95, '\*')+"\n")

    # MACHINE EFFICIENCY

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F01'")

    sum\_down1 = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F01'")

    sum\_run1 = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F02'")

    sum\_down2 = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F02'")

    sum\_run2 = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F03'")

    sum\_down3 = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F03'")

    sum\_run3 = c.fetchone()[0]

    f01\_eff = '{0:.0f}%'.format((sum\_down1/sum\_run1)\*100)

    f02\_eff = '{0:.0f}%'.format((sum\_down2/sum\_run2)\*100)

    f03\_eff = '{0:.0f}%'.format((sum\_down3/sum\_run3)\*100)

    # MACHINE TROUBLE

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F01' AND downtime\_factor = 'Machine Trouble'")

    sum\_down1\_machine\_trb = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F01'")

    sum\_run1\_machine\_trb = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F02' AND downtime\_factor = 'Machine Trouble'")

    sum\_down2\_machine\_trb = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F02'")

    sum\_run2\_machine\_trb = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F03' AND downtime\_factor = 'Machine Trouble'")

    sum\_down3\_machine\_trb = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F03'")

    sum\_run3\_machine\_trb = c.fetchone()[0]

    f01\_machine\_trb = '{0:.0f}%'.format((sum\_down1\_machine\_trb/sum\_run1\_machine\_trb)\*100)

    f02\_machine\_trb = '{0:.0f}%'.format((sum\_down2\_machine\_trb/sum\_run2\_machine\_trb)\*100)

    f03\_machine\_trb = '{0:.0f}%'.format((sum\_down3\_machine\_trb/sum\_run3\_machine\_trb)\*100)

    # QUALITY PROBLEM

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F01' AND downtime\_factor = 'Quality Problem'")

    sum\_down1\_quality = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F01'")

    sum\_run1\_quality = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F02' AND downtime\_factor = 'Quality Problem'")

    sum\_down2\_quality = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F02'")

    sum\_run2\_quality = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F03' AND downtime\_factor = 'Quality Problem'")

    sum\_down3\_quality = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F03'")

    sum\_run3\_quality = c.fetchone()[0]

    f01\_quality = '{0:.0f}%'.format((sum\_down1\_quality/sum\_run1\_quality)\*100)

    f02\_quality = '{0:.0f}%'.format((sum\_down2\_quality/sum\_run2\_quality)\*100)

    f03\_quality = '{0:.0f}%'.format((sum\_down3\_quality/sum\_run3\_quality)\*100)

    # CLEANING

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F01' AND downtime\_factor = 'Cleaning'")

    sum\_down1\_cleaning = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F01'")

    sum\_run1\_cleaning = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F02' AND downtime\_factor = 'Cleaning'")

    sum\_down2\_cleaning = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F02'")

    sum\_run2\_cleaning = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F03' AND downtime\_factor = 'Cleaning'")

    sum\_down3\_cleaning = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F03'")

    sum\_run3\_cleaning = c.fetchone()[0]

    f01\_cleaning = '{0:.0f}%'.format((sum\_down1\_cleaning/sum\_run1\_cleaning)\*100)

    f02\_cleaning = '{0:.0f}%'.format((sum\_down2\_cleaning/sum\_run2\_cleaning)\*100)

    f03\_cleaning = '{0:.0f}%'.format((sum\_down3\_cleaning/sum\_run3\_cleaning)\*100)

    # CLEANING

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F01' AND downtime\_factor = 'Breakdown'")

    sum\_down1\_breakdown = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F01'")

    sum\_run1\_breakdown = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F02' AND downtime\_factor = 'Breakdown'")

    sum\_down2\_breakdown = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F02'")

    sum\_run2\_breakdown = c.fetchone()[0]

    c.execute("SELECT sum(downtime) FROM machine\_data WHERE machine\_num = 'F03' AND downtime\_factor = 'Breakdown'")

    sum\_down3\_breakdown = c.fetchone()[0]

    c.execute("SELECT sum(runtime) FROM machine\_data WHERE machine\_num = 'F03'")

    sum\_run3\_breakdown = c.fetchone()[0]

    f01\_breakdown = '{0:.0f}%'.format((sum\_down1\_breakdown/sum\_run1\_breakdown)\*100)

    f02\_breakdown = '{0:.0f}%'.format((sum\_down2\_breakdown/sum\_run2\_breakdown)\*100)

    f03\_breakdown = '{0:.0f}%'.format((sum\_down3\_breakdown/sum\_run3\_breakdown)\*100)

    f01\_nonprod = '{0:.0f}%'.format(100-((sum\_down1/sum\_run1)\*100))

    f02\_nonprod = '{0:.0f}%'.format(100-((sum\_down2/sum\_run2)\*100))

    f03\_nonprod = '{0:.0f}%'.format(100-((sum\_down3/sum\_run3)\*100))

    all\_data = [['Machine','Machine Eff','Machine Trb','Quality Prob','Cleaning','Breakdown','NonProd'],

                ['F01',f01\_eff,f01\_machine\_trb,f01\_quality,f01\_cleaning,f01\_breakdown,f01\_nonprod],

                ['F02',f02\_eff,f02\_machine\_trb,f02\_quality,f02\_cleaning,f02\_breakdown,f02\_nonprod],

                ['F03',f03\_eff,f03\_machine\_trb,f03\_quality,f03\_cleaning,f03\_breakdown,f03\_nonprod]

                ]

    from tabulate import tabulate

    print(tabulate(all\_data,headers='firstrow'))

    print("\n"+footer.center(95, '\*'))

def defective\_ratio():

    header = " DEFECTIVE RATIO REPORT "

    footer = " END OF REPORT "

    print("\n"+header.center(95, '\*')+"\n")

    c.execute("SELECT sum(total\_qty) FROM lot\_information WHERE biscuit\_type = 'Chocolate'")

    choco\_ttl = c.fetchone()[0]

    c.execute("SELECT sum(defect\_qty) FROM quality\_data as A INNER JOIN lot\_information as B ON A.lot\_num = B.lot\_num WHERE biscuit\_type = 'Chocolate'")

    choco\_defect = c.fetchone()[0]

    c.execute("SELECT sum(total\_qty) FROM lot\_information WHERE biscuit\_type = 'Plain'")

    plain\_ttl = c.fetchone()[0]

    c.execute("SELECT sum(defect\_qty) FROM quality\_data as A INNER JOIN lot\_information as B ON A.lot\_num = B.lot\_num WHERE biscuit\_type = 'Plain'")

    plain\_defect = c.fetchone()[0]

    choco\_def\_ratio = '{0:.0f}%'.format((choco\_defect/choco\_ttl)\*100)

    plain\_def\_ratio = '{0:.0f}%'.format((plain\_defect/plain\_ttl)\*100)

    all\_data = [['Biscuit Type','Total Qty','Defect Qty','Defective Ratio'],

                ['Chocolate',choco\_ttl,choco\_defect,choco\_def\_ratio],

                ['Plain',plain\_ttl,plain\_defect,plain\_def\_ratio]

    ]

    from tabulate import tabulate

    print(tabulate(all\_data,headers='firstrow'))

    print("\n"+footer.center(95, '\*'))

main\_menu()