# My NEA – M&S Schedule Planner

Contents

[My NEA – M&S Schedule Planner 1](#_Toc130382359)

**[Reasearch:](#_Toc130382360)** [4](#_Toc130382360)

[Overview on the problem: 4](#_Toc130382361)

[Why I chose to do this as my NEA: 5](#_Toc130382362)

[Some limitations to my NEA: 5](#_Toc130382363)

[Who will be the third party for my NEA: 5](#_Toc130382364)

[Interview: 6](#_Toc130382365)

[Questionnaire: 6](#_Toc130382366)

[Current Situation: 7](#_Toc130382367)

[Summary: 8](#_Toc130382368)

**[Analysis:](#_Toc130382369)** [9](#_Toc130382369)

[IPSO: 9](#_Toc130382370)

[Background to the problem: 9](#_Toc130382371)

[The current system: 9](#_Toc130382372)

[The potential users: 10](#_Toc130382373)

[The users needs: 10](#_Toc130382374)

[The data: 10](#_Toc130382375)

[Data dictionary: 11](#_Toc130382376)

[Flowchart: 11](#_Toc130382377)

[Analasys: 16](#_Toc130382378)

[UserInterface 17](#_Toc130382379)

[Add Employee Window 20](#_Toc130382380)

[Add Holiday Window 23](#_Toc130382381)

[Add Shift Window 25](#_Toc130382382)

[Add Window 28](#_Toc130382383)

[Database controller 30](#_Toc130382384)

[Delete Employee Window 35](#_Toc130382385)

[Delete Holiday Window 37](#_Toc130382386)

[Delete Shift Window 39](#_Toc130382387)

[Delete Window 41](#_Toc130382388)

[Scheduler 43](#_Toc130382389)

[View Employees Window 47](#_Toc130382390)

[View Holiday Window 48](#_Toc130382391)

[View Schedule Window 49](#_Toc130382392)

[View Window 50](#_Toc130382393)

[Window 52](#_Toc130382394)

# **Reasearch:**

## Overview on the problem:

With my NEA I try to prouce a product that can be used at my part time job (Marks and Spencers). What I would be looking to do is to try and make a schedule planner for all the employees in a certain part of the business. For this project I am going to focus on the clothing and home apsect of the store in which I work in. I think that it would be beneficial to the business as now, they are using drawn out tables with peoples names and their times of breaks. Plus the majority of the time, there is writing on this timetable that has been printed out to try and help make sure that none of the breaks clash and it all runs smoothly. The reason that I decided to make this is because I think that it would be a lot more efficient than the system that is in place now. This would lead to the managers having more free time in the morning which will mean that that time could be utilised in other ways.

In this work schedule planner, I will include:

• The name of the employee,

• The time In which they start their shift,

• The time in which they end their shift,

• The times in which they are on their break,

• The amount of time that they are allowed to be on their break for,

• The department in which they are working in (either : Mens clothing, womens clothing, childrens clothing and Home wear.)

After I have inputted this data, I will be able to make a program that will be able to make a schedule. This schedule will help the business massively as it would help save them a large amount of time, that could be used doing something more productive. This means that there is an actual use for the business.

## Why I chose to do this as my NEA:

The reason in which I chose to do this for my NEA is because I thought it would be useful for the business. This is because I see my managers struggle to make a schedule every morning that I am working and they are stood there for a good 30 minuets, trying to figure out how they can make the schedule work for the day. Plus, I only work on the weekends, so you could imagine that they do this every day trying to figure it out. This costs them a lot of time that they could be using in a better way, but they are stuck there trying to make it. So I think that if I was to make my NEA work, then I think that it would be greatly beneficial for them and for me. As in the time that they are trying to figure out the schedule, they could be on the shop floor helping customers and making use of their time. This would help the whole business out in my opinion.

Another reason that I am looking to do this is that the managers making it themselves means that they are prone to making mistakes. As humans we are bound to make mistakes. Occasionally, they make it so that our breaks overlap so there isnt enough people on the floor for someone to go onto break. This means that we do not even stick to the schedule that they make us. This sometimes means that they have wasted their time and it does not work as well as It should. But if I was to make it be generated completely through the computer, that would mean that there would not be any overlaps on the breaks and every one can stick to the breaks that have been distributed to them.

## Some limitations to my NEA:

There arent many limitations to my NEA but there can be some. One of them is that if there is an employee helping a customer and it is time for them to go onto their break, they would not be able to go on the time that they have given to them and they would have to go whenever they have finished helping the customer. This will create a domino effect and will effect everyone elses breaks. This could be an issue that I would not be able to fix. This is because it is very unpredictable for when people are likely to go on their breaks or not. This is an issue that is out of my control and I cannot do anything about it. This is something that we are just going to work around as a business and we would have to communicate with eachother if our breaks were to overun and we were to go late on our break.

## Who will be the third party for my NEA:

I have chosen my manager to be the third party and one of the employees to be the third parties for my NEA.

The reason that I chose to use my manager as one of the third parties is that this whole NEA is to try and solve a problem that they are facing and it will be useful for them. This would save them a lot of time.

Also, the reason that I chose to have an employee to overview my NEA is because they would be able to tell me first hand how well the system that I put into place will work and how efficient and practical it is. They will be folowing the schedule that I am making so they are bale to tell me how good it works and what needs improvement.

## Interview:

I spoke to one of the employees that work with me and I asked them about the problems with the system that is in place now. They were telling me about how every so often that the breaks clash and it can cause there to bee too little people on the shop floor helping customers and it can be overwhelming for the few emplyees on the floor while the other employees are away on break and not on the floor. She also was telling me about how there isnt really a practical system that is in place now and the employees usually must communicate on when they start and end their shifs and when they are going on their break. This made me think that I should make a solution to this problem and figure out a way in which I could change the system that is in place right now and make it better and more practical. This is what lead me to do my NEA in the first place.

Here will be a few questions that I asked them and what they responded with:

## Questionnaire:

Question 1: How well do you think the system that is in place at M&S is working for employees?

I think the system we have in place does work….. once every couple months. We never stick to it and its like there isnt a point of having one. It is just wasting the managers time and they cant use it in a more productive way.

Question 2: Why do you think that this system is not working?

I think that the reason the system doesn’t work is because its mainly human made. This is because they do not have a machiene that could just make it for them. If they had that it would be so much easier.

Question 3: What challenges are faced with the system that Is in place at the moment?

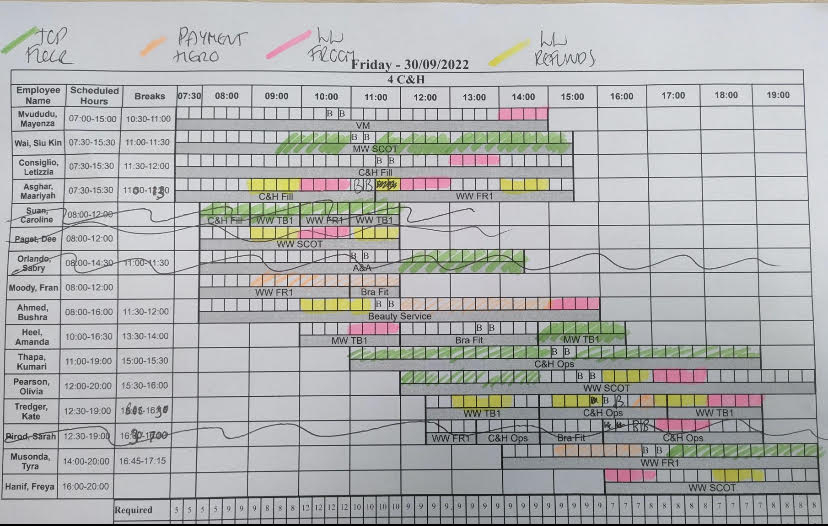
I think that because it Is the managers making the time table thing, it means that a lot of the time, there are breaks that clash and that leads to confusion and there not being enough people on the floor at that time. Also if a customer that is speaking to an employee when it is time for them to go onto break, then that means that their break is gone on later than usual and it means that everyones break is later than it should be. This cases them to be more confused on when they should be going and not going. These are the most ustrating thing that keeps happening. Its easily avoidable aswell.

Question 4: How well do you think that the computer generated timetable would work?

I actually think that that would work much better than the system that we have in place at the moment. This is because all of the human error that keeps happening. The managers start writing out the timetable for the day, and then something distracts them, and then they just forget and lose their train of thought. This means that when they come back to finsihing It, they forget what theyre doing and mess up the timetable. This leads to fustration from the employees and the managers are forced to spend more time on trying to fix it.

## Current Situation:

This is the current timetable that they have:



As you can see from the image above, there is a lot of human input that has to go into the schedule for the day. Once again this is very time consuming and there is many problems with the schedule. On the schedule, you can see the scheduled hours that they are contracted to work, the time of breaks that they have, and there is a little table with the times at which they are working in different parts of the floor. This is indicated by the colours on the schedule. Green means that they are on the top floor, which is mens, kids and home. Light orange is payment Hero where there is an employee who Is flexable enoungh to be moving from the top and bottom floor on the tills. They go onto whatever floor is busiest at that point in time. Pink is for when those spesific employees need to be working in the womens wear section on that day. They need to be on the bottom floor and working in the womens wear section if it is highlighted pink. Finally, green is for womens wear tills. This is when there are a couple of employees by the tills that stay there and help customers use the self service tills. They aslo help them with refunds and exchanges if there is any. There are also other jobs that need to be done like ISF. This is where there is an employee that goes around the store and picks up all of the items that have been ordered for click and collect. This is usually put on the timetable but on this day it has not been put on. This is once again quite confusing and it means that not everyone knows where theyre meant to be at all times. Another problem with this timetable is that there is also blank spaces on the people that are in at that point in time but they do not have a place that they need to be working. This means that they do not know what they need to be doing and where they should go. This causes large amounts of confusion on behalf of both the employees and the managers. As you can also see on the timetable, there are certain people that have been crossed out. This is because they have called in sick and they are unable to work on that day. This is quite a difficult task to automate as it is near impossible to predict when or if someone is going to be ill on that spesific day. But this means that the managers have to re go through the timetable and make sure that the amount of people on each floor is proportional to eachother. This

## Summary:

The problem that we are facing at my workplace is that there is a lack of a system for when employees can go onto break and when they start and end their shift. This means that employees can become confused and must figure out themselves when they need to go on their break and when they need to stay on the shop floor. There is also confusion on which part of the business that employees are working at. So my NEA will mean that there is a better system in place and employees will not be confused when the come into the workplace and they will know what is going on. This will help prevent any problems with their being a lack of employees on the shop floor. This will decrease the workload on them and they will not be as stressed.

# **Analysis:**

## IPSO:

|  |  |
| --- | --- |
| **IN** | **Process** |
| **NameOfEmployees**  **StartTimeOfShift**  **EndTimeOfShift**  **AmountOfTimeForBreak**  **WhichSectionTheyAreWorkingIn** | **If shifts clash?**  **If breaks clash?**  **How many people working on a spesific day?**  **If there is enough people that start at the opening time and closing time working?** |
| **Store** | **Out** |
| **NameOfEmployees**  **StartTimeOfShift**  **EndTimeOfShift**  **AmountOfTimeForBreak**  **Schedule** | **Schedule** |

## Background to the problem:

I first noticed to the problem when I was about a month into working there. This is when I realised

how bad the schedule that we have in place at the moment is compared to what it could be. It first

happened when it was one Saturday when I was working and there was a lot of confusion that was

going around due to someone going on their break later than they should have. This meant that

everyones breaks were pushed back. The made me aware of the problem due to the fact that not

everyone knew that therer would have to be a delay in everyones break. This then lead me to start

thinking of ways that I could change the schedule. But I thought that there was no reason for me to

make a new schedule. Then I was told that we had to do the NEA, and this problem was the first

thing that came to my mind. This lead me to choose this to be the focus of the NEA.

## The current system:

The system that is in place at the moment is not very efficient. As shown in the picture above, there

is a large amount of human input that goes into them making the schedule. This usually causes there

to be problems. This leads to them having to change the shifts and the breaks mid way through the

day. This leads to ther ebeing confusions around the workplace due to the managers not being able

to go around to every employee and tell them what has changed. This means that some people

know about the new changes and some others don’t. this means that there are some people going

off of the new schedule that has been made and then some people going off of the old system. This

can be very bad for the business as sometimes the new schedule that is put in place during the day

can be wildly different to the one that they started off with on that day. This means that there are

people in the wrong places at the wrong times. This then causes there to be miscommunication in

the workplace and then there are gaps in places that need to have employees.

## The potential users:

The main users of my NEA will be my managers. This is due to them needing to make a schedule

every morning for the rest of the employees. This means that they will be uning my NEA every day

that they are working (which is most days of the year as they will need the schedule every day the

shop is open). This also means that the employees will see the schedule every morning that they are

working. This means that there will be a lot of use of my NEA for both the employees and the

managers. This will stop the confusion for both of them.

## The users needs:

The needs that the mangers have is for the schedule to:

~ show the names of each employee working on that day.

~ show all of the times that the employees shifts start and end.

~ show the times scheduled for the employees breaks for when they need to go on their break and

come back from it.

~ what area of the shop that each employee has to be working in at that time.

~ whether they are in on time or not.

## The data:

There will be a lot of data that goes into the schedule. This means that I will have to get the data

from the managers or on teams. At M&amp;S, we have a teams group chat that has all of the people that

work in the clothing and home section in the business. This means that I can get their names and the

times that they start and end their shifts and on what days that they work. This means that I have

access to all of the data that I will need to use to make it so that the schedule works. This means that

there is no added data that I need to go look for as I already have access to all of it.

## Data dictionary:

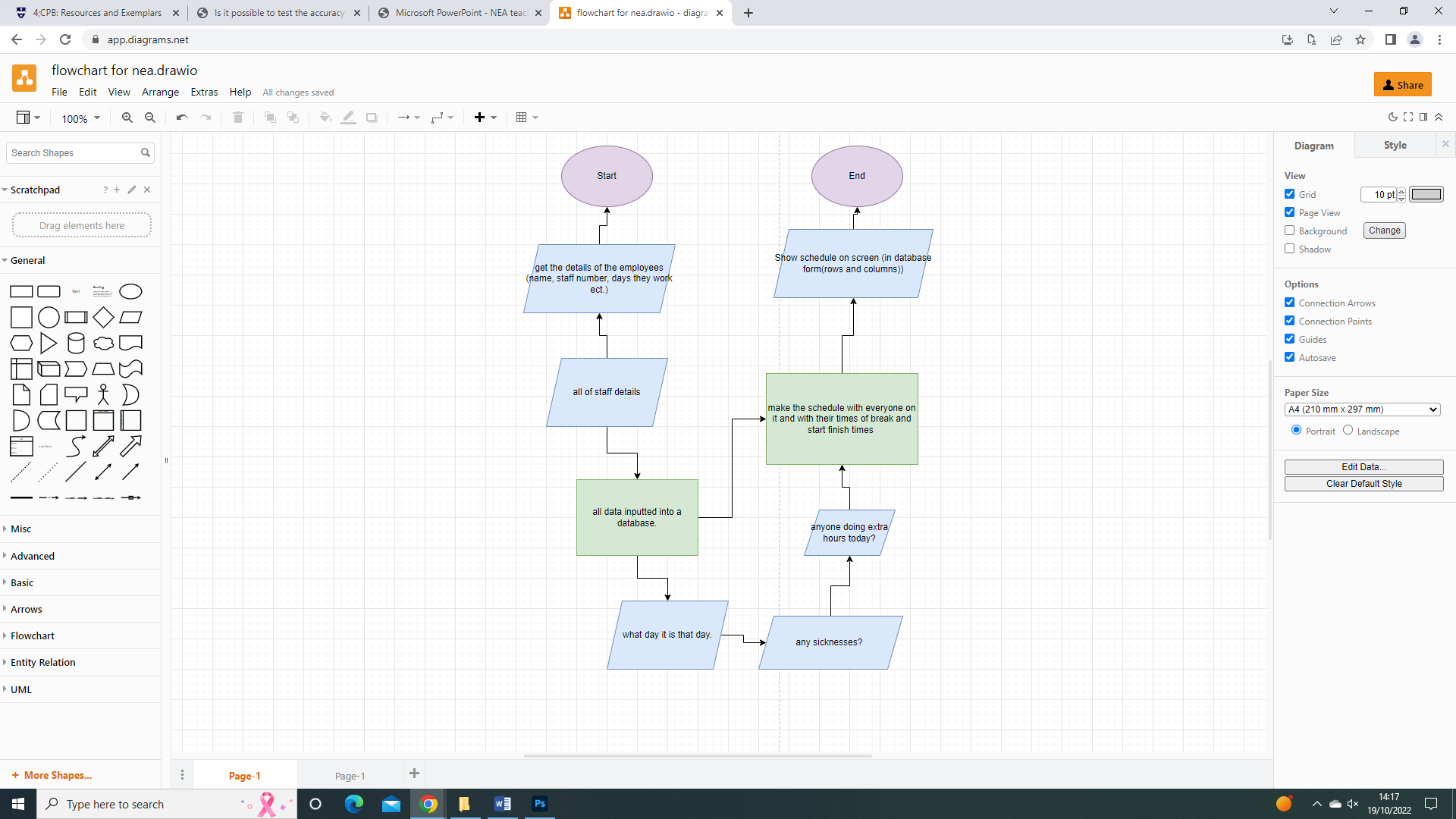
A data dictionary is A Data Dictionary is a collection of names, definitions, and attributes about data

elements that are being used or captured in a database, information system, or part of a research

project. With this prohject all of the data that I will need, I already have access to through teams,

this means that I will not have to go looking for more data and I have all of it accessible to me.

## Flowchart:



**Requirements list**

1. To be easy to use.
   1. the user must be able to input the data needed with ease
   2. the user must be returned with the schedule fully complete
   3. the schedule must not need any more added human input after done

To be able to store the employees data.

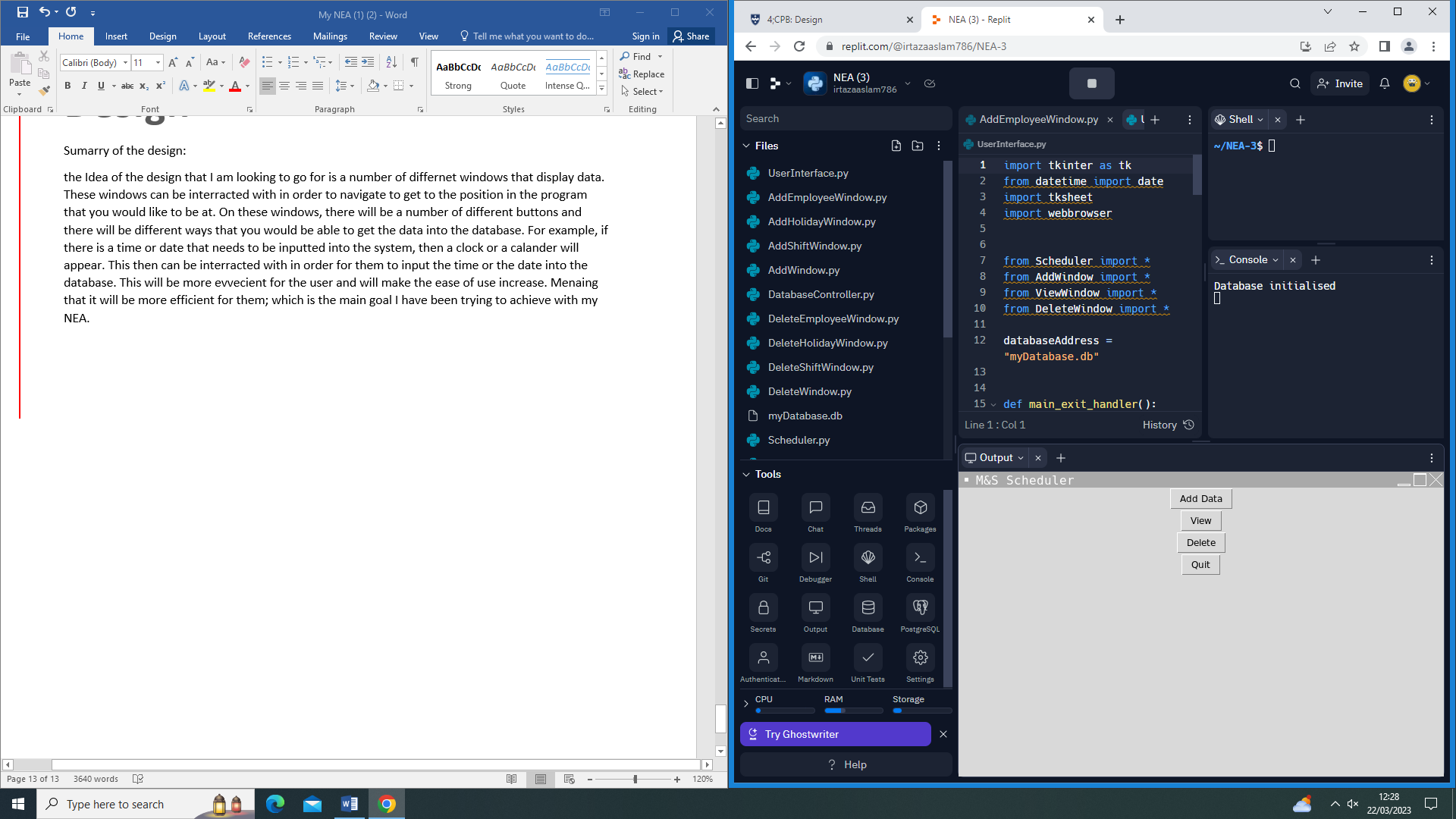
* 1. the end user must be able to access the database and be able to configure it
  2. the user must be able to make changes to the database
  3. be able to add and delete elements into the database.
  4. Be able to make changes if the data inputted is wrong.
  5. All data must be in the correct data type and not have to change the data if something were to be inputted wrong.

1. To be able to store all of the data about the employees working times and days.
   1. The user must be able to access this and be able to make changes if needed.
2. To be able to make the schedule
   1. The system must be able to make the schedule where there is no overlapping in shift times and breaks throughout the day.
   2. The system will have to be able to retrive the data from the database and then work out how many people should be working on each day and at which times.
   3. The schedule must then be outputted with all of the data that the data base has on each of the employees and with all of the start and finish times.
3. The schedule to be clear
   1. The schedule needs to be in a clear format so it is easier for everyone to understand
   2. The employees should not have to look at the schedule for a long time for them to have to understand it; they should be able to understand it very quickly.
4. Schedule to be made efficiently.
   1. After the user has entered the data, they should not be required to re enter it into the database even if the program is closed.
   2. If all the data has been entered, the user will still have an option to add more data and remove data if there was to be a new employee, or an employee left.
   3. If no data is needed to be entered and all of the data entered is all that the user needs to use, then the user would have to load the program, enter the day that it is on that occasion, if there has been any sickness calls from employees then the schedule should be made in a short amount of time and it to be clear for everyone to be able to read.

**Design**

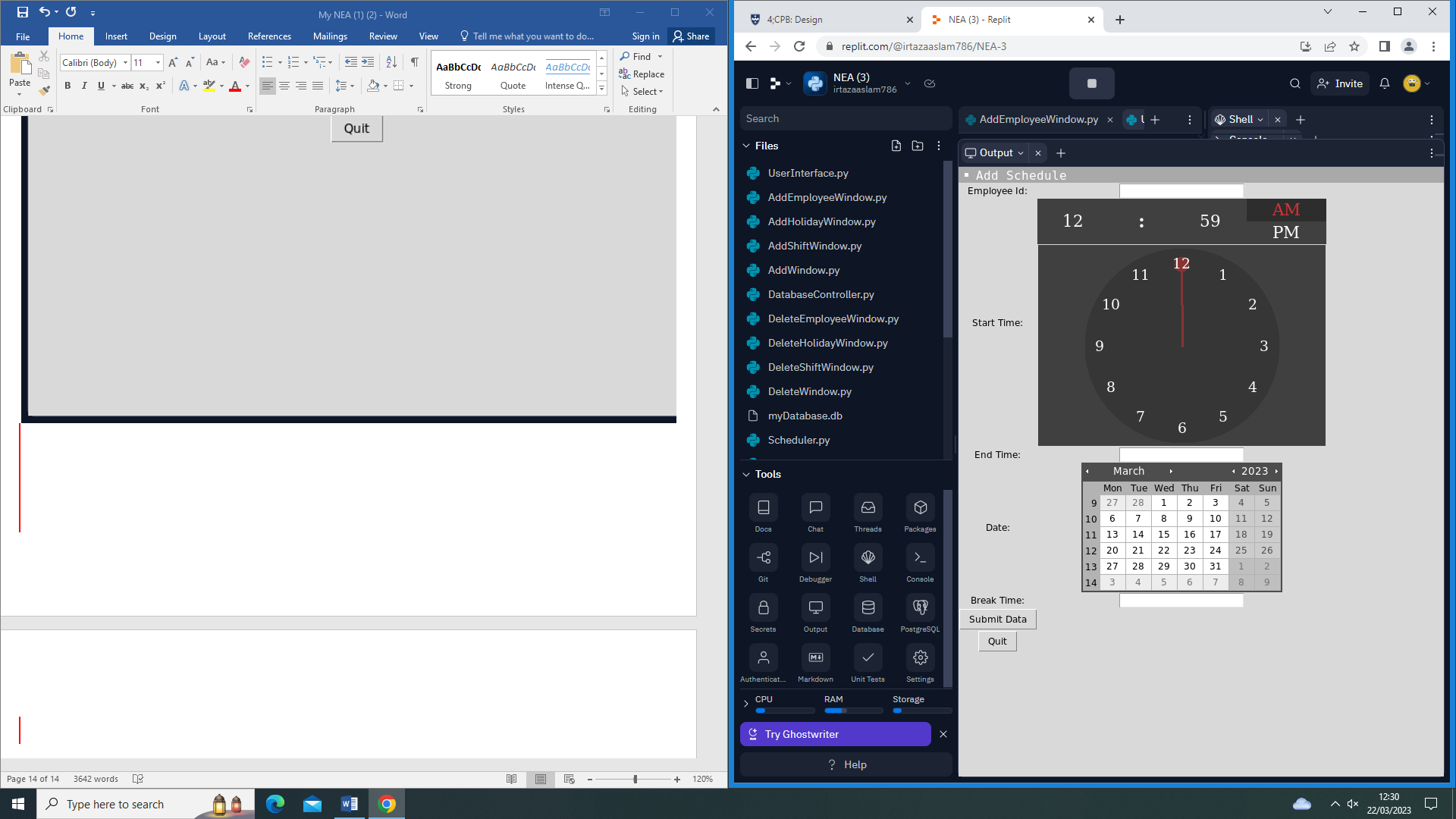
Sumarry of the design:

the Idea of the design that I am looking to go for is a number of differnet windows that display data. These windows can be interracted with in order to navigate to get to the position in the program that you would like to be at. On these windows, there will be a number of different buttons and there will be different ways that you would be able to get the data into the database. For example, if there is a time or date that needs to be inputted into the system, then a clock or a calander will appear. This then can be interracted with in order for them to input the time or the date into the database. This will be more evvecient for the user and will make the ease of use increase. Menaing that it will be more efficient for them; which is the main goal I have been trying to achieve with my NEA.

Main Page:

This Is the start page that all of the users will start on. This is where they will navigate their way through the system in order to get to the position that they would want to be in.

Page to add data:



This is the page in which you can add a shift to the schedule. This is where the calander and the clock are required in order to make it easier for the managers.

# Analasys:

As you can see in the images above, these are the ways in which that the managers are able to input the data and navigate the system. This is set out in a very clear way in order for the user to not become confused when using the system. This will mean that they are able to input data in an easier way. Also, on each of the windows, there are headers at the top of the window that show what part of the program that you are at. This makes it easier for the user to understand where they are in the system and they will be able to nagivate more efficiently.

In this project I will be using SQLite3 , TKSHEET and Tkinter. The reason for me choosing these three are because:

I chose SQLite3 because I thought that it would be the most efficient way to make a database that I am able to input and take data out with. There are many different ways to utilise SQLite3 to move data, add data, take data out ect. This is the reason why I chose SQLite3.

I chose to use Tkinter and Tksheet as they both are cruicial tools in order to show the data in a nice format. These both can be used to create new windows and these are what could be used to input the data into the database and also output the data in a neater format. This will make it easier for the user to understand the data and It will make it far easier to display the data.

The whole project is split up into individual classes. Each of these classes are in a new file. This means that in the majority of the files, i have imported data from another file in order for it to run. The reason that I did this was because I thought that it would make the code look neater and it will easier to understand from an outside perspective. All of the files work hand in hand with eachother so If there is a problem with any parts of the code, then I would be able to tell just by running the code as the code will not run and it will give me an error back. Some may believe that it is not a good idea to make all of the classes in different files but I believe that it made it easier for me so I know where all the bits of code are at a time.

With the program, the user would be able to input the data in through the simplified data entry that will then take the data into a database. This data will be stored in the database and can be used on a later date. This means that the user does not need to input data every time they log on to use the system; once again more efficient for them. This data in the database will be displayed on a table that they user will be able to see with all of their data clearly showing in the table.

**Technical Solution**

# **UserInterface**

import tkinter as tk

from datetime import date

import tksheet

from Scheduler import \*

from AddWindow import \*

from ViewWindow import \*

from DeleteWindow import \*

databaseAddress = "myDatabase.db"

def main\_exit\_handler():

root.destroy()

scheduler.closeDatabase()

def showWindow(oldWindow):

root.deiconify()

oldWindow.destroy()

def StartWindow():

startwindow = StartWindow(root, "1200x800", scheduler)

def createAddWindow():

addWindow = AddWindow(root, "1200x800", scheduler)

def createViewWindow():

viewWindow = ViewWindow(root, "1200x800", scheduler)

def createDeleteWindow():

deleteWindow = DeleteWindow(root, "1200x800", scheduler)

def closeProgram():

root.destroy()

scheduler.closeDatabase()

root = tk.Tk()

root.protocol("WM\_DELETE\_WINDOW", main\_exit\_handler)

root.title("M&S Scheduler")

root.geometry('1200x800')

scheduler = Scheduler(databaseAddress)

addDataButton = tk.Button(root, text="Add Data", command=createAddWindow)

ViewButton = tk.Button(root, text="View", command=createViewWindow)

DeleteButton = tk.Button(root, text="Delete", command=createDeleteWindow)

QuitButton = tk.Button(root, text="Quit", command=closeProgram)

addDataButton.pack()

ViewButton.pack()

DeleteButton.pack()

QuitButton.pack()

root.mainloop()

# Add Employee Window

import tkinter as tk

from tkinter import messagebox

from Window import \*

class AddEmployeeWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Add Employee")

self.addFirstNameWidgets()

self.addSurnameWidgets()

self.addAgeWidgets()

self.addDepartmentWidgets()

self.submitDataButton()

self.QuitButton.grid(row = 5, column = 0)

def addFirstNameWidgets(self):

self.firstNameLabel = tk.Label(self.window, text="First name:")

self.firstNameLabel.grid(row = 0, column = 0)

self.firstNameEntry = tk.Entry(self.window)

self.firstNameEntry.grid(row = 0, column = 1)

def addSurnameWidgets(self):

self.surNameLabel = tk.Label(self.window, text="Surname:")

self.surNameLabel.grid(row = 1, column = 0)

self.surNameEntry = tk.Entry(self.window)

self.surNameEntry.grid(row = 1, column = 1)

def addAgeWidgets(self):

self.ageLabel = tk.Label(self.window, text="Age:")

self.ageLabel.grid(row = 2, column = 0)

self.ageEntry = tk.Entry(self.window)

self.ageEntry.grid(row = 2, column = 1)

def addDepartmentWidgets(self): #TODO: make dropdown box

self.departmentLabel = tk.Label(self.window, text="Department:")

self.departmentLabel.grid(row = 3, column = 0)

self.departmentEntry = tk.Entry(self.window)

self.departmentEntry.grid(row = 3, column = 1)

def submitDataButton(self):

self.submitDataButton = tk.Button(self.window, text="Submit Data", command=self.getData)

self.submitDataButton.grid(row = 4, column = 0)

def getData(self):

try:

age = int(self.ageEntry.get())

except:

messagebox.showinfo("Invalid Data", "Please check your input, you have entered invalid data!")

return

firstName = self.firstNameEntry.get()

surName = self.surNameEntry.get()

department = self.departmentEntry.get()

successfulEntry = self.scheduler.addEmployee(firstName, surName, age, department)

if not successfulEntry:

messagebox.showinfo("Invalid Data", "Please check your input, you have entered invalid data!")

else:

self.firstNameEntry.delete(0, 'end')

self.surNameEntry.delete(0, 'end')

self.ageEntry.delete(0, 'end')

self.departmentEntry.delete(0, 'end')

# Add Holiday Window

import tkinter as tk

from tkinter import messagebox

from tkcalendar import Calendar, DateEntry

from Window import \*

class AddHolidayWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Add Holiday")

self.addEmployeeIDWidgets()

self.addStartDateWidgets()

self.addEndDateWidgets()

self.submitDataButton()

self.QuitButton.grid(row = 4, column = 0)

def addEmployeeIDWidgets(self):

self.employeeIDLabel = tk.Label(self.window, text="Employee ID:")

self.employeeIDLabel.grid(row = 0, column = 0)

self.employeeIDEntry = tk.Entry(self.window)

self.employeeIDEntry.grid(row = 0, column = 1)

def addStartDateWidgets(self):

self.startDateLabel = tk.Label(self.window, text="Start date:")

self.startDateLabel.grid(row = 1, column = 0)

self.startDatePicker = Calendar(self.window, selectmode = 'day', year = 2023, month = 3, date\_pattern="dd/mm/yyyy")

self.startDatePicker.grid(row = 1, column = 1)

def addEndDateWidgets(self):

self.endDateLabel = tk.Label(self.window, text="End date:")

self.endDateLabel.grid(row = 2, column = 0)

self.endDatePicker = Calendar(self.window, selectmode = 'day', year = 2023, month = 3, date\_pattern="dd/mm/yyyy")

self.endDatePicker.grid(row = 2, column = 1)

def submitDataButton(self):

self.submitDataButton = tk.Button(self.window, text="Submit Data", command=self.getData)

self.submitDataButton.grid(row = 3, column = 0)

def getData(self):

employeeID = self.employeeIDEntry.get()

startDate = self.startDatePicker.get\_date()

endDate = self.endDatePicker.get\_date()

successfulEntry = self.scheduler.addHoliday(employeeID, startDate, endDate)

if not successfulEntry:

messagebox.showinfo("Invalid Data", "Please check your input, you have entered invalid data")

else:

self.employeeIDEntry.delete(0, 'end')

# Add Shift Window

import tkinter as tk

from tkinter import messagebox

from Window import \*

from tkcalendar import Calendar

from tktimepicker import AnalogPicker, AnalogThemes

class AddShiftWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Add Schedule")

self.addEmployeeIdWidget()

self.addStartTimeWidget()

self.addEndTimeWidget()

self.addBreakTimeWidget()

self.addDateWidget()

self.CreateSubmitDataButton()

self.QuitButton.grid(row=6, column=0)

def addEmployeeIdWidget(self):

self.EmployeeIdLabel = tk.Label(self.window, text="Employee Id:")

self.EmployeeIdLabel.grid(row=0, column=0)

self.EmployeeIdEntry = tk.Entry(self.window)

self.EmployeeIdEntry.grid(row=0, column=1)

def addStartTimeWidget(self):

self.StartTimeLabel = tk.Label(self.window, text="Start Time:")

self.StartTimeLabel.grid(row=1, column=0)

self.StartTimePicker = AnalogPicker(self.window)

theme = AnalogThemes(self.StartTimePicker)

theme.setDracula()

self.StartTimePicker.grid(row=1, column=1)

def addEndTimeWidget(self):

self.EndTimeLabel = tk.Label(self.window, text="End Time:")

self.EndTimeLabel.grid(row=2, column=0)

self.EndTimeEntry = tk.Entry(self.window)

self.EndTimeEntry.grid(row=2, column=1)

def addDateWidget(self):

self.DateLabel = tk.Label(self.window, text="Date:")

self.DateLabel.grid(row=3, column=0)

self.DatePicker = Calendar(self.window, selectmode='day', year=2023, month=3, date\_pattern="dd/mm/yyyy")

self.DatePicker.grid(row=3, column=1)

def addBreakTimeWidget(self):

self.BreakTimeLabel = tk.Label(self.window, text="Break Time:")

self.BreakTimeLabel.grid(row=4, column=0)

self.BreakTimeEntry = tk.Entry(self.window)

self.BreakTimeEntry.grid(row=4, column=1)

def CreateSubmitDataButton(self):

self.submitDataButton = tk.Button(self.window, text="Submit Data", command=self.getData)

self.submitDataButton.grid(row=5, column=0)

def formatTime(self, hours, minutes, period):

if period == 'PM':

hours += 12

elif hours < 10:

return '0' + str(hours) + ":" + str(minutes)

return str(hours) + ":" + str(minutes)

def getData(self):

minutes = self.StartTimePicker.minutes()

hours = self.StartTimePicker.hours()

period = self.StartTimePicker.period()

print(self.formatTime(hours, minutes, period))

'''

employeeID = self.EmployeeIdEntry.get()

startTime = 0

endTime = self.EndTimeEntry.get()

date = self.DatePicker.get\_date()

breakTime = self.BreakTimeEntry.get()

successfulEntry = self.scheduler.addShift(employeeID, startTime, endTime, date, breakTime)

if not successfulEntry:

messagebox.showinfo("Invalid Data", "Please check your input, you have entered invalid data!")

else:

self.EmployeeIdEntry.delete(0, 'end')

self.StartTimeEntry.delete(0, 'end')

self.EndTimeEntry.delete(0, 'end')

self.BreakTimeEntry.delete(0, 'end')'''

# Add Window

import tkinter as tk

from Window import \*

from AddEmployeeWindow import \*

from AddShiftWindow import \*

from AddHolidayWindow import \*

class AddWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Add Data")

self.createAddEmployeeButton()

self.createAddShiftButton()

self.createAddHolidayButton()

self.QuitButton.pack()

def createAddEmployeeButton(self):

self.employeeButton = tk.Button(self.window, text="Add Employee", command=self.createAddEmployeeWindow)

self.employeeButton.pack()

def createAddShiftButton(self):

self.shiftButton = tk.Button(self.window, text="Add Shift", command=self.createAddShiftWindow)

self.shiftButton.pack()

def createAddHolidayButton(self):

self.holidayButton = tk.Button(self.window, text="Add Holiday", command=self.createAddHolidayWindow)

self.holidayButton.pack()

def createAddEmployeeWindow(self):

self.addEmployeeWindow = AddEmployeeWindow(self.window, self.geometry, self.scheduler)

def createAddShiftWindow(self):

self.addShiftWindow = AddShiftWindow(self.window, self.geometry, self.scheduler)

def createAddHolidayWindow(self):

self.addHolidayWindow = AddHolidayWindow(self.window, self.geometry, self.scheduler)

# Database controller

import sqlite3

CHECK\_IF\_EMPLOYEES\_EXISTS = "SELECT name FROM sqlite\_master WHERE type='table' AND name='Employees'"

CHECK\_IF\_SHIFTS\_EXISTS = "SELECT name FROM sqlite\_master WHERE type='table' AND name='Shifts'"

CHECK\_IF\_HOLIDAYS\_EXISTS = "SELECT name FROM sqlite\_master WHERE type='table' AND name='Holidays'"

class DatabaseController():

def \_\_init\_\_(self, filePath):

try:

self.connection = sqlite3.connect(filePath)

print("Database initialised")

self.connection.execute("PRAGMA foreign\_keys = ON")

cursor = self.connection.cursor()

cursor.execute(CHECK\_IF\_EMPLOYEES\_EXISTS)

if cursor.fetchall() == []: #Employees table does not exist

self.createEmployeesTable()

print("Employees table successfully created")

cursor.execute(CHECK\_IF\_SHIFTS\_EXISTS)

if cursor.fetchall() == []: #Shifts table does not exist

self.createShiftsTable()

print("Shifts table successfully created")

cursor.execute(CHECK\_IF\_HOLIDAYS\_EXISTS)

if cursor.fetchall() == []: #Shifts table does not exist

self.createHolidaysTable()

print("Holidays table successfully created")

cursor.close()

except sqlite3.Error as error:

print("Error initialising database")

print(error)

def executeSelectQuery(self, query):

cursor = self.connection.cursor()

results = cursor.execute(query)

resultSet = []

for row in results:

resultSet.append(row)

#each row is a tuple

cursor.close()

return resultSet

def executeQuery(self, query):

success = True

try:

cursor = self.connection.cursor()

cursor.execute(query)

self.connection.commit()

except sqlite3.Error as error:

print(error)

success = False

finally:

cursor.close()

return success

def closeDatabase(self):

self.connection.close()

print("Database closed")

def createEmployeesTable(self):

try:

self.connection.execute("""CREATE TABLE Employees(

EmployeeID integer primary key,

firstName varchar(20) NOT NULL,

surname varchar(20) NOT NULL,

age integer NOT NULL,

department varchar(20) NOT NULL

)""")

except sqlite3.Error as error:

print(error)

self.closeDatabase()

def createEmployee(self, firstName, surname, age, department):

query = "INSERT INTO Employees (firstName, surname, age, department) VALUES (\""

query += firstName + "\", \"" + surname + "\", " + str(age) + ", \"" + department + "\");"

return self.executeQuery(query)

def getColumnNames(self, query):

cursor = self.connection.execute(query)

return [data[0] for data in cursor.description]

def createShiftsTable(self):

try:

self.connection.execute("""CREATE TABLE Shifts(

ShiftID integer primary key,

EmployeeID integer NOT NULL,

StartTime varchar(5) NOT NULL,

endTime varchar(5) NOT NULL,

date varchar(10) NOT NULL,

breakTime varchar(5) NOT NULL,

CONSTRAINT fk\_employee

FOREIGN KEY (EmployeeID)

REFERENCES Employees(EmployeeID)

ON DELETE CASCADE

)""")

except sqlite3.Error as error:

print(error)

self.closeDatabase()

# times: "HH:MM" (5 characters)

# Date : "DD/MM/YYYY" (10 characters)

def createShift(self, employeeID, startTime, endTime, date, breakTime):

query = "INSERT INTO Shifts (employeeID, startTime, endTime, date, breakTime) VALUES ("

query += str(employeeID) + ", \"" + startTime + "\", \"" + endTime + "\", \"" + date + "\", \"" +breakTime+ "\");"

return self.executeQuery(query)

def createHolidaysTable(self):

try:

self.connection.execute("""CREATE TABLE Holidays(

HolidayID integer primary key,

EmployeeID integer NOT NULL,

startDate varchar(10) NOT NULL,

endDate varchar(10) NOT NULL,

CONSTRAINT fk\_employee

FOREIGN KEY (EmployeeID)

REFERENCES Employees(EmployeeID)

ON DELETE CASCADE

)""")

except sqlite3.Error as error:

print(error)

self.closeDatabase()

def createHoliday(self, employeeID, startDate, endDate):

query = "INSERT INTO Holidays (EmployeeID, startDate, endDate) VALUES ("

query += str(employeeID) + ", \"" + startDate + "\", \"" + endDate + "\");"

return self.executeQuery(query)

# Delete Employee Window

import tkinter as tk

from tkinter import messagebox

from Window import \*

class DeleteEmployeeWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Delete Employee")

self.DeleteFirstNameWidget()

self.DeleteSurNameWidget()

self.DeleteDepartmentWidget()

self.submitDataButton()

self.QuitButton.grid(row = 4, column = 0)

def DeleteFirstNameWidget(self):

self.FirstNameLabel = tk.Label(self.window, text="Firstname:")

self.FirstNameLabel.grid(row = 0, column = 0)

self.FirstNameEntry = tk.Entry(self.window)

self.FirstNameEntry.grid(row = 0, column = 1)

def DeleteSurNameWidget(self):

self.SurNameLabel = tk.Label(self.window, text="Surname:")

self.SurNameLabel.grid(row = 1, column = 0)

self.SurNameEntry = tk.Entry(self.window)

self.SurNameEntry.grid(row = 1, column = 1)

def DeleteDepartmentWidget(self):

self.DepartmentLabel = tk.Label(self.window, text="Department:")

self.DepartmentLabel.grid(row = 2, column = 0)

self.DepartmentEntry = tk.Entry(self.window)

self.DepartmentEntry.grid(row = 2, column = 1)

def submitDataButton(self):

self.submitDataButton = tk.Button(self.window, text="Submit Data", command=self.getData)

self.submitDataButton.grid(row = 3, column = 0)

def getData(self):

firstname = self.FirstNameEntry.get()

surname = self.SurNameEntry.get()

department = self.DepartmentEntry.get()

successful = self.scheduler.deleteEmployee(firstname, surname, department)

if not successful:

messagebox.showinfo("Invalid Data", "Please check your input, you have entered invalid data!")

else:

self.FirstNameEntry.delete(0, 'end')

self.SurNameEntry.delete(0, 'end')

self.DepartmentEntry.delete(0, 'end')

# Delete Holiday Window

import tkinter as tk

from tkinter import messagebox

from DeleteWindow import \*

class DeleteHolidayWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Delete Holiday")

self.DeleteFirstNameWidget()

self.DeleteSurNameWidget()

self.DeleteStartDateWidget()

self.CreateSubmitDataButton()

self.QuitButton.grid(row = 4, column = 0)

def DeleteFirstNameWidget(self):

self.FirstNameLabel = tk.Label(self.window, text="Firstname:")

self.FirstNameLabel.grid(row = 0, column = 0)

self.FirstNameEntry = tk.Entry(self.window)

self.FirstNameEntry.grid(row = 0, column = 1)

def DeleteSurNameWidget(self):

self.SurNameLabel = tk.Label(self.window, text="Surname:")

self.SurNameLabel.grid(row = 1, column = 0)

self.SurNameEntry = tk.Entry(self.window)

self.SurNameEntry.grid(row = 1, column = 1)

def DeleteStartDateWidget(self):

self.StartDateLabel = tk.Label(self.window, text="StartDate:")

self.StartDateLabel.grid(row = 2, column = 0)

self.StartDateEntry = tk.Entry(self.window)

self.StartDateEntry.grid(row = 2, column = 1)

def CreateSubmitDataButton(self):

self.submitDataButton = tk.Button(self.window, text="Submit Data", command=self.getData)

self.submitDataButton.grid(row = 3, column = 0)

def getData(self):

firstname = self.FirstNameEntry.get()

surname = self.SurNameEntry.get()

startdate = self.StartDateEntry.get()

successful = self.scheduler.deleteHoliday(firstname, surname, startdate)

if not successful:

messagebox.showinfo("Invalid Data", "Please check your input, you have entered invalid data!")

else:

self.FirstNameEntry.delete(0, 'end')

self.SurNameEntry.delete(0, 'end')

self.StartDateEntry.delete(0, 'end')

# Delete Shift Window

import tkinter as tk

from tkinter import messagebox

from Window import \*

class DeleteShiftWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Delete Shift")

self.DeleteFirstNameWidget()

self.DeleteSurNameWidget()

self.DeleteStartTimeWidget()

self.DeleteEndTimeWidget()

self.CreateSubmitDataButton()

self.QuitButton.grid(row = 5, column = 0)

def DeleteFirstNameWidget(self):

self.FirstNameLabel = tk.Label(self.window, text="First Name :")

self.FirstNameLabel.grid(row=0, column=0)

self.FirstNameEntry = tk.Entry(self.window)

self.FirstNameEntry.grid(row=0, column=1)

def DeleteSurNameWidget(self):

self.SurNameLabel = tk.Label(self.window, text="Surname:")

self.SurNameLabel.grid(row=1, column=0)

self.SurNameEntry = tk.Entry(self.window)

self.SurNameEntry.grid(row=1, column=1)

def DeleteStartTimeWidget(self):

self.StartTimeLabel = tk.Label(self.window, text="StartTime:")

self.StartTimeLabel.grid(row=2, column=0)

self.StartTimeEntry = tk.Entry(self.window)

self.StartTimeEntry.grid(row=2, column=1)

def DeleteEndTimeWidget(self):

self.EndTimeLabel = tk.Label(self.window, text="End Time:")

self.EndTimeLabel.grid(row=3, column=0)

self.EndTimeEntry = tk.Entry(self.window)

self.EndTimeEntry.grid(row=3, column=1)

def CreateSubmitDataButton(self):

self.submitDataButton = tk.Button(self.window, text="Submit Data", command=self.getData)

self.submitDataButton.grid(row=4, column=0)

def getData(self):

firstname = self.FirstNameEntry.get()

surname = self.SurNameEntry.get()

starttime = self.StartTimeEntry.get()

endtime = self.EndTimeEntry.get()

successfulDeletion = self.scheduler.deleteShift(firstname, surname, starttime, endtime)

if not successfulDeletion:

messagebox.showinfo("Invalid Data", "Please check your input, you have entered invalid data!")

else:

self.FirstNameEntry.delete(0, 'end')

self.SurNameEntry.delete(0, 'end')

self.StartTimeEntry.delete(0, 'end')

self.EndTimeEntry.delete(0, 'end')

# Delete Window

import tkinter as tk

from Window import \*

from DeleteShiftWindow import \*

from DeleteEmployeeWindow import \*

from DeleteHolidayWindow import \*

class DeleteWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("Delete Employees")

self.createDeleteEmployeeButton()

self.createDeleteShiftButton()

self.createDeleteHolidayButton()

self.QuitButton.pack()

def createDeleteEmployeeButton(self):

self.employeeButton = tk.Button(self.window, text="Delete Employee", command=self.createDeleteEmployeeWindow)

self.employeeButton.pack()

def createDeleteShiftButton(self):

self.shiftButton = tk.Button(self.window, text="Delete Shift", command=self.createDeleteShiftWindow)

self.shiftButton.pack()

def createDeleteHolidayButton(self):

self.holidayButton = tk.Button(self.window, text="Delete Holiday", command=self.createDeleteHolidayWindow)

self.holidayButton.pack()

def createDeleteEmployeeWindow(self):

self.DeleteEmployeeWindow = DeleteEmployeeWindow(self.window, self.geometry, self.scheduler)

def createDeleteShiftWindow(self):

self.DeleteShiftWindow = DeleteShiftWindow(self.window, self.geometry, self.scheduler)

def createDeleteHolidayWindow(self):

self.DeleteHolidayWindow = DeleteHolidayWindow(self.window, self.geometry, self.scheduler)

# Scheduler

from DatabaseController import \*

class Scheduler:

def \_\_init\_\_(self, filename):

self.DBcontroller = DatabaseController(filename)

def getShiftsForDay(self, date):

query = """SELECT Employees.firstName, Employees.surname, Shifts.startTime, Shifts.endTime, Shifts.breakTime

FROM Shifts JOIN Employees

ON Shifts.EmployeeID = Employees.EmployeeID

WHERE Shifts.date = \"""" + date + "\";"

return self.DBcontroller.executeSelectQuery(query), self.DBcontroller.getColumnNames(query)

def addEmployee(self, firstName, surname, age, department):

if len(firstName) > 20 or len(firstName) == 0:

return False

elif len(surname) > 20 or len(surname) == 0:

return False

elif age < 17:

return False

elif len(department) > 20 or len(department) == 0:

return False

return self.DBcontroller.createEmployee(firstName, surname, age, department)

def addShift(self, employeeID, startTime, endTime, date, breakTime):

if len(startTime) != 5:

return False

elif len(endTime) != 5:

return False

elif len(date) != 10:

return False

elif len(breakTime) != 5:

return False

return self.DBcontroller.createShift(employeeID, startTime, endTime, date, breakTime)

def addHoliday(self, employeeID, startDate, endDate):

return self.DBcontroller.createHoliday(employeeID, startDate, endDate)

def closeDatabase(self):

self.DBcontroller.closeDatabase()

def getEmployeeData(self):

query = """SELECT \* from Employees;"""

return self.DBcontroller.executeSelectQuery(query), self.DBcontroller.getColumnNames(query)

def getHolidays(self):

query = """SELECT Holidays.HolidayID, Employees.firstName, Employees.surname, Holidays.startDate, Holidays.endDate

FROM Holidays JOIN Employees

ON Holidays.EmployeeID = Employees.EmployeeID

ORDER BY Holidays.startDate ASC;"""

return self.DBcontroller.executeSelectQuery(query), self.DBcontroller.getColumnNames(query)

def deleteEmployee(self, firstName, surname, department):

query = """SELECT EmployeeID

FROM Employees

WHERE firstName = \"{0}\" AND surname = \"{1}\" AND department = \"{2}\""""

query = query.format(firstName, surname, department)

employeeID = self.DBcontroller.executeSelectQuery(query)[0][0]

query = "DELETE FROM Employees WHERE EmployeeID = " + str(employeeID)

return self.DBcontroller.executeQuery(query)

def deleteShift(self, firstName, lastName, startTime, endTime):

query = """SELECT Shifts.ShiftID

FROM Shifts JOIN Employees ON Employees.EmployeeID = Shifts.ShiftID

WHERE Employees.firstName = \"{0}\" AND Employees.surname = \"{1}\" AND Shifts.startTime = \"{2}\" AND Shifts.endTime = \"{3}\""""

query = query.format(firstName, lastName, startTime, endTime)

shiftID = self.DBcontroller.executeSelectQuery(query)[0][0]

query = "DELETE FROM Shifts WHERE ShiftID = " + str(shiftID)

return self.DBcontroller.executeQuery(query)

def deleteHoliday(self, firstName, surname, startDate):

query = """SELECT Holidays.HolidayID

FROM Employees JOIN Holidays ON Employees.EmployeeID = Holidays.EmployeeID

WHERE Employees.firstName = \"{0}\" AND Employees.surname = \"{1}\" AND Holidays.startDate = \"{2}\""""

query = query.format(firstName, surname, startDate)

holidayID = self.DBcontroller.executeSelectQuery(query)[0][0]

query = "DELETE FROM Holidays WHERE HolidayID = " + str(holidayID)

return self.DBcontroller.executeQuery(query) import tkinter as tk

import tksheet

from Window import \*

class ViewEmployeesWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("View Employees")

self.createEmployeesTable()

self.table.pack()

self.QuitButton.pack()

def createEmployeesTable(self):

self.table = tksheet.Sheet(self.window, width = 300, height = 200) #make dimensions dynamic

employeeData, columns = self.scheduler.getEmployeeData()

self.table.headers(columns)

self.table.set\_sheet\_data(employeeData)

# View Employees Window

import tkinter as tk

import tksheet

from Window import \*

class ViewEmployeesWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("View Employees")

self.createEmployeesTable()

self.table.pack()

self.QuitButton.pack()

def createEmployeesTable(self):

self.table = tksheet.Sheet(self.window, width = 300, height = 200) #make dimensions dynamic

employeeData, columns = self.scheduler.getEmployeeData()

self.table.headers(columns)

self.table.set\_sheet\_data(employeeData)

# View Holiday Window

import tkinter as tk

import tksheet

from Window import \*

class ViewHolidaysWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("View Holidays")

self.createHolidaysTable()

self.table.pack()

self.QuitButton.pack()

def createHolidaysTable(self):

self.table = tksheet.Sheet(self.window, width = 300, height = 200) #make dimensions dynamic

holidayData, columns = self.scheduler.getHolidays()

self.table.headers(columns)

self.table.set\_sheet\_data(holidayData)

# View Schedule Window

import tkinter as tk

import tksheet

from Window import \*

class ViewScheduleWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("View Schedule")

self.createScheduleTable()

self.table.pack()

self.QuitButton.pack()

def createScheduleTable(self):

self.table = tksheet.Sheet(self.window, width = 300, height = 200) #make size dynamic

shiftData, columns = self.scheduler.getShiftsForDay(self.getDate())

self.table.headers(columns)

self.table.set\_sheet\_data(shiftData)

# View Window

import tkinter as tk

from Window import \*

from ViewScheduleWindow import \*

from ViewEmployeesWindow import \*

from ViewHolidaysWindow import \*

class ViewWindow(Window):

def \_\_init\_\_(self, master, geometry, scheduler):

super().\_\_init\_\_(master, geometry, scheduler)

self.window.title("View Data")

self.createSchedulerButton()

self.createEmployeesButton()

self.createHolidayButton()

self.QuitButton.pack()

def createSchedulerButton(self):

self.scheduleButton = tk.Button(self.window, text="View Schedule", command=self.createViewScheduleWindow)

self.scheduleButton.pack()

def createViewScheduleWindow(self):

viewScheduleWindow = ViewScheduleWindow(self.window, self.geometry, self.scheduler)

def createEmployeesButton(self):

self.employeesButton = tk.Button(self.window, text="View Employees", command=self.createViewEmployeesWindow)

self.employeesButton.pack()

def createHolidayButton(self):

self.holidayButton = tk.Button(self.window, text="View Holidays", command=self.createViewHolidayWindow)

self.holidayButton.pack()

def createViewHolidayWindow(self):

viewHolidayWindow = ViewHolidaysWindow(self.window, self.geometry, self.scheduler)

def createViewEmployeesWindow(self):

viewEmployeesWindow = ViewEmployeesWindow(self.window, self.geometry, self.scheduler)

# Window

import tkinter as tk

from datetime import datetime

class Window:

def \_\_init\_\_(self, master, geometry, scheduler):

self.master = master

self.scheduler = scheduler

self.geometry = geometry

self.master.withdraw()

self.window = tk.Toplevel(self.master)

self.window.geometry(geometry)

self.window.protocol("WM\_DELETE\_WINDOW", self.closeWindow)

self.QuitButton = tk.Button(self.window, text="Quit", command=self.closeWindow)

def closeWindow(self):

self.master.deiconify()

self.window.destroy()

def getDate(self):

return datetime.today().strftime('%d/%m/%Y')

##### Evaluation:

In conclusion, I believe that the project that I have created is fit for pupose and easy to use. One of my main requirements for this project was to make a easy to use, efficient system that the user can use in order to schedule out the day of work ahead. I believe that I have achieved this. My NEA is easy to use and efficient and outputs data at the speed that is required. My NEA does everything that it needed to achieve and extra.

One thing that I would change about my NEA is the start screen. At the moment, when you enter into the system, then you are put straight onto a menu where you can choose what you want to do. I would have prefered to implement a start screen into my code. I feel like I could have implemented this if I had more time and could have been one of the improvments I added to my system. But this would have hindered the efficiency of the code which was the main goal that I was going for with this program.

Another thing that I would have changed about my NEA is if there is not a employee in on a spesific day. If an employee is on holiday on a particiular day, then I would want my project to be able to assign the shift to someone else. This would be quite difficult to implement into the code due to the set days that employees already work. This will mean that the project would not be able to assign the shift to any of the people.

Plus, another thing that I would have added to my NEA if I had more time is a sick day widget. This would be where when an employee is ill and does not come in on a spesific day, then the manager will be able to show the schedule without them on it. If there is a lack of employees that day due to holidays or sicknesses, this is when the program goes out to try and find another employee they can slot into that shift. This would mean that there would be no problems with the schedule that is given back to the user.

Finally, another thing that I could have added could have been break time allocation. This is where the program assings spesific times for people to go on their breaks. This will mean that there is not a problem with the breaks overriding and the users would not have to waste time inputting all of the breaks into the system. This will cause the project to be even more efficint and even easier to use.

Overall I am happy with how my project turned out due to the fact that it does what I wanted to create. The code works as planned and there are not any issues that I have to deal with in regards to the project.