Timetable Generation

Contents Page No:

Introduction 2

Software and Hardware requirements of the project. 2

Block diagram of the system 3

List of functionalities being implemented 4

Correlation of these components with OS concepts 4

Which Algorithms adopted from OS concepts studies? Why? Why not other alternatives if any? 5

Flowchart and Code 6-15

Limitation and Future scope 15

List of References. 16

Introduction

A mini system has been built for timetable generation. The user inputs the total weekly requirement of four different course branches for four different faculties. These parameters have been set since this is just a simulator. It can very easily be scaled up or modified according to the user requirements.

Software and Hardware requirements of the project

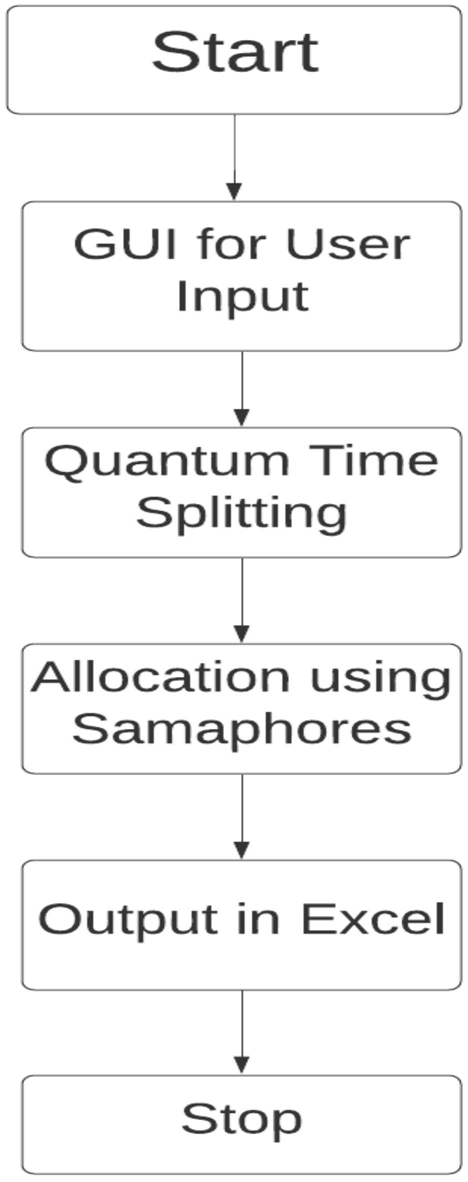
SOFTWARE REQUIREMENTS

- Operating system – Windows 10 is used as the operating system as it is stable and supports more features and is more user friendly.
- Development tools and Programming language – Python is used to write the whole code and Python Tkinter is used for designing and styling of front end. PyCharm IDE has been used but any other IDE or text editor can also be used. Python libraries used are tkinter (for GUI) and openpyxl (for excel file manipulation).
- Spreadsheet application – MS Excel has been used to store and display the timetable generated.

HARDWARE REQUIREMENTS

- Intel core i7 8th generation is used as a processor. Intel core i5 should also suffice.
- 8 GB RAM is used. Minimum of 2 GB RAM is advisable.
- Intel core i3 with 1GB RAM can also be used.

Block diagram of the system :



List of functionalities being implemented

- Generate a timetable for all the branches according to their respective requests.
- Ensure that no faculty has consecutive lectures in a particular branch but is evenly distributed throughout the entire week.
- Ensure that no faculty has been allotted more than one branch in one time slot.

Correlation of these components with OS concepts

- Process Management – In this project, the different branches (B Tech CS, B Tech IT etc.) act as the processes competing for resources (in this case the resource is memory in the form faculty time)
- File Management – The only file management taking place is in the form of excel file creation for the timetable. As such there is no need to use a particular algorithm since the system is not dealing with multiple files here. It is made sure that the timetable generated previously for different set of input values does not affect other generations.
- Thread efficient utilization of resources – For the efficient utilization of memory (faculty time), the process (total requirement of a branch for a particular faculty in hours) is broken down into threads (total requirement in hours is split into quantum of 1 hour using the quantum_time_splitter code).
- Concurrency – The different branches are being allocated faculty time in parallel.
- Deadlock handling – A semaphore algorithm has been designed to ensure that no deadlocks occur i.e. no two branches will be allocated the same faculty for a particular time slot.
- Starvation prevention – To ensure that one branch doesn’t use up long durations of time of a particular faculty, a quantum time splitter algorithm has been used which switches the allocation to another branch after the quantum time expires (in this case 1 hour of faculty time).

Which Algorithms adopted from OS concepts studied? Why? Why not other alternatives if any?

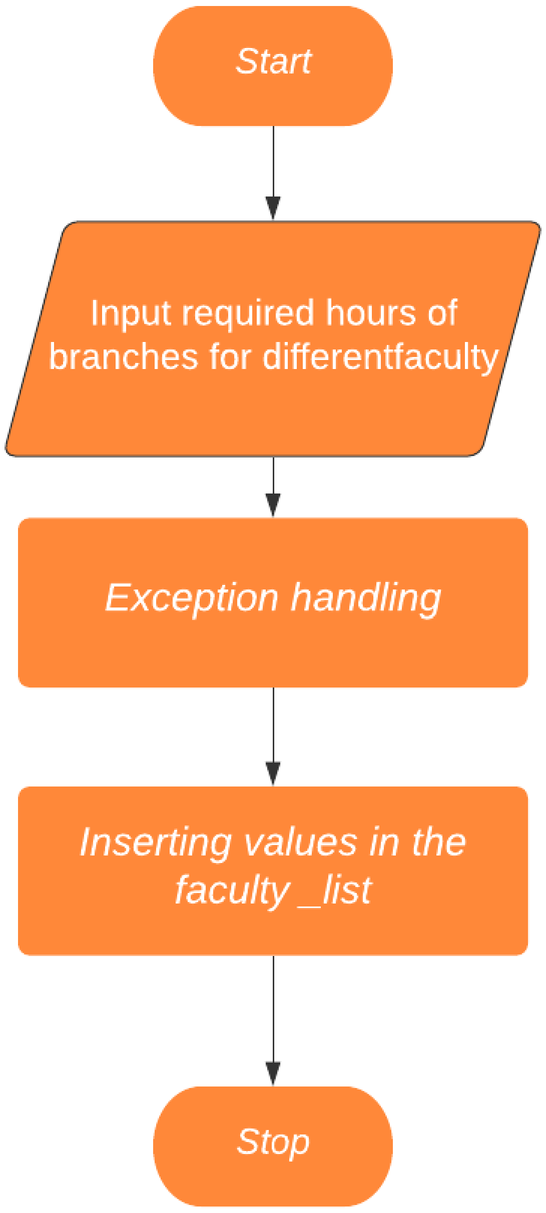
- Semaphores – Semaphore is a very significant technique to manage concurrent processes by using a simple integer value. This variable is used to solve the critical section problem and to achieve process synchronization in the multiprocessing environment. In our project, semaphores are used to ensure that no deadlocks occur i.e. no two branches will be allocated the same faculty for a particular time slot. If a particular time slot of a faculty has not been allocated, then '-1' is used to represent it.
- Time Quantum – The period of time for which a process is allowed to run in a pre-emptive multitasking system is generally called the time slice or quantum. The scheduler is run once every time slice to choose the next process to run. In our project, a quantum of 1 hour is used while allocating faculty time to a particular branch. This ensures that one branch doesn't use up long durations of time of a particular faculty due to which other branches might suffer.

Alternatives: In place of semaphores, monitors or message passing can also be used, but in this case the requirements of the functionality designed is more than fulfilled using semaphores. The simplicity of semaphores is also an added advantage.

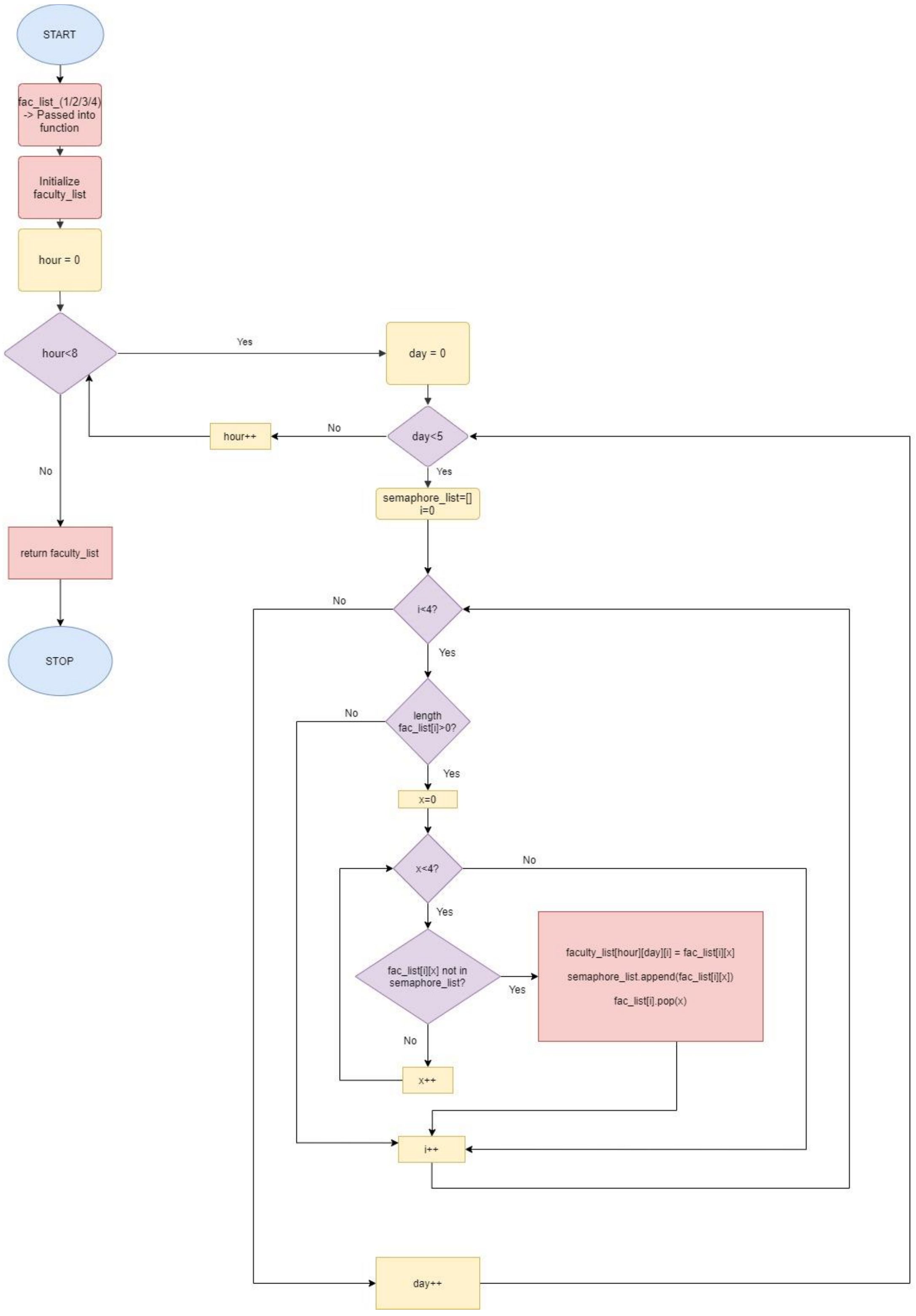
Flowchart and Code :

Flowcharts:

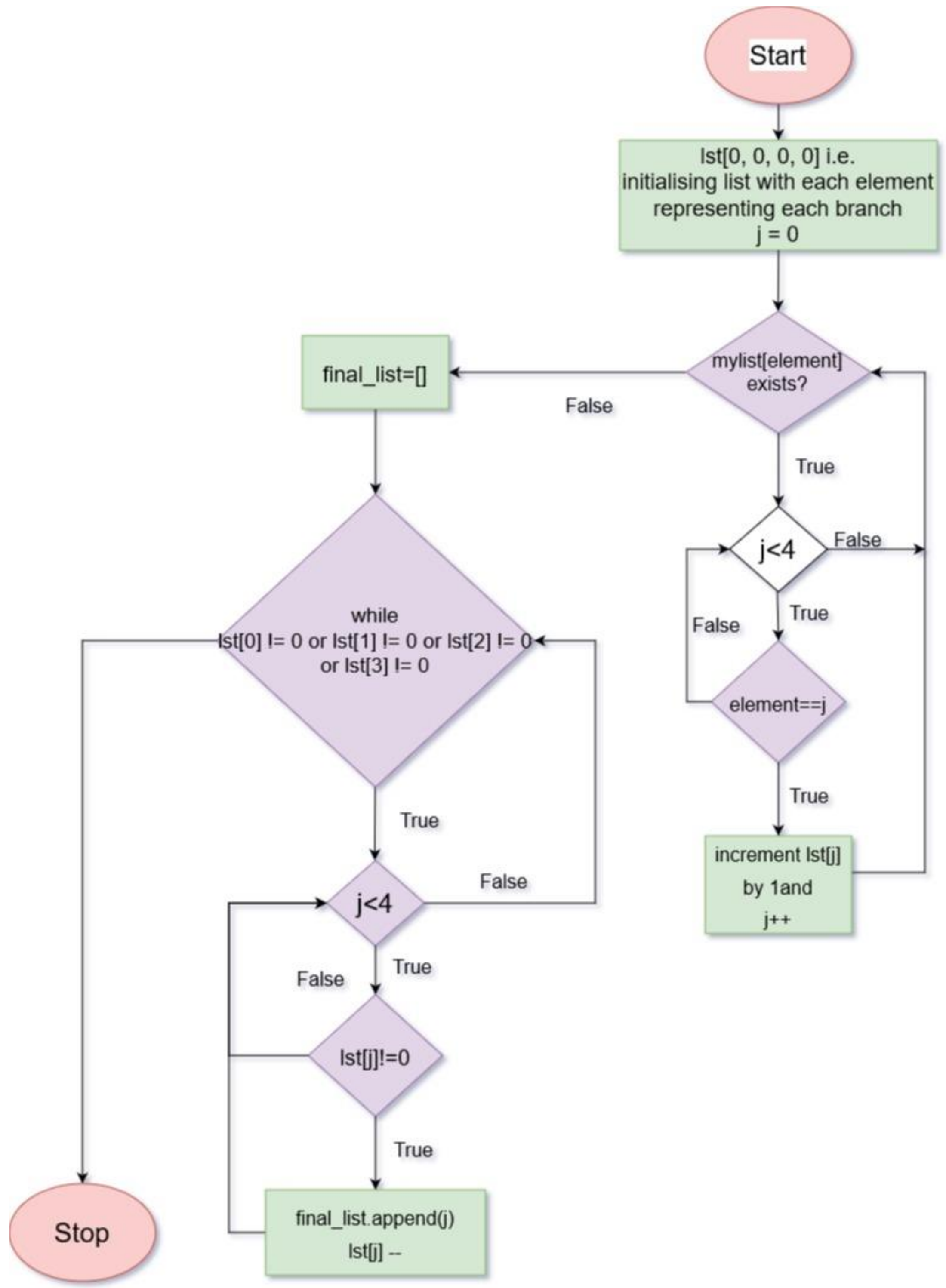
GUI for User Input:


Start
Input required hours of branches for differentfaculty
Exception handling
Inserting values in the faculty _list
Stop

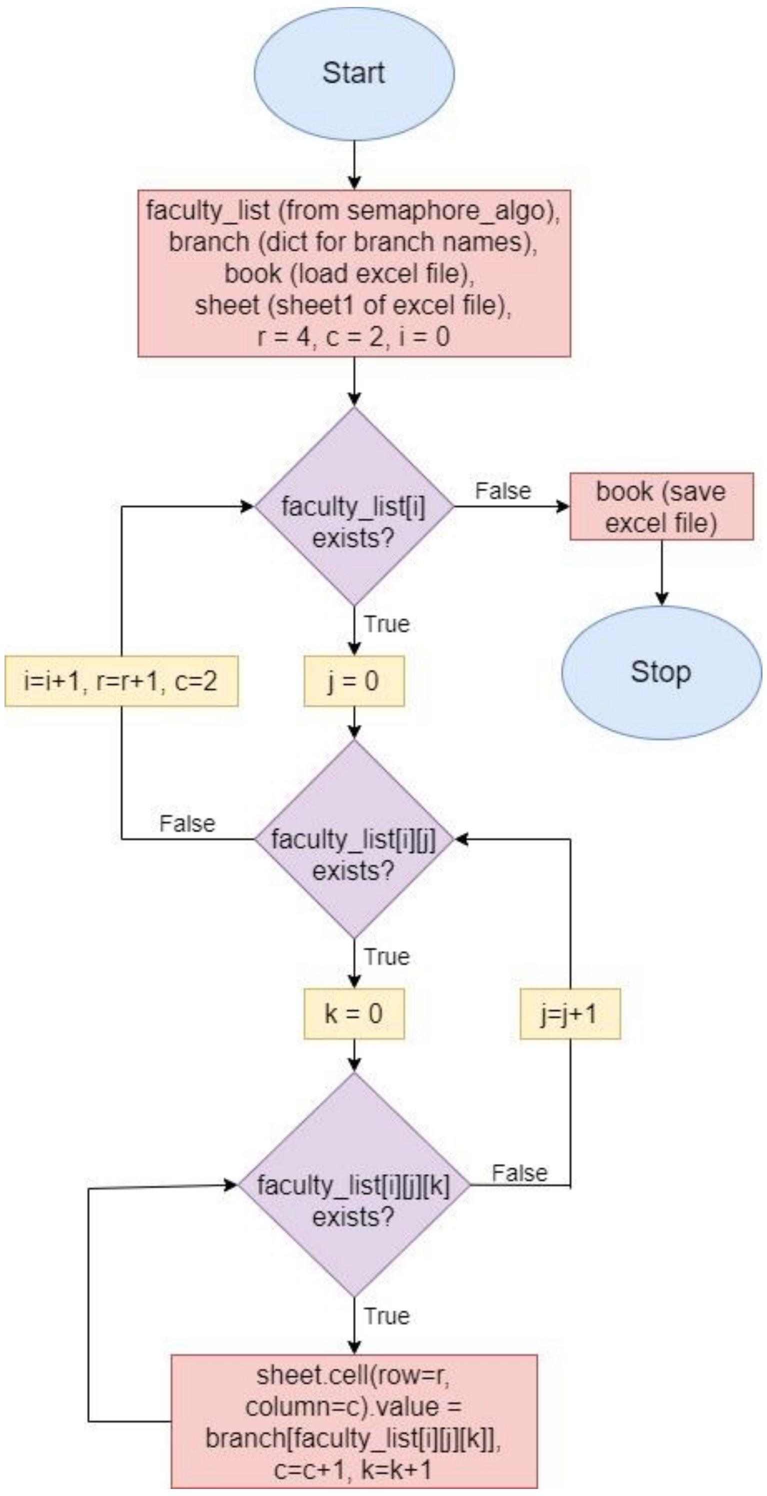
Semaphores:


START
fac_list_(1/2/3/4) -> Passed into function
Initialize faculty_list
hour = 0
hour<8
Yes
day = 0
day<5
No
hour++
No
Yes
semaphore_list=[] i=0
return faculty_list
STOP
i<4?
No
Yes
length fac_list[i]>0?
No
Yes
x=0
x<4?
No
Yes
fac_list[i][x] not in semaphore_list?
Yes
faculty_list[hour][day][i] = fac_list[i][x]
semaphore_list.append(fac_list[i][x])
fac_list[i].pop(x)
No
x++
i++
day++

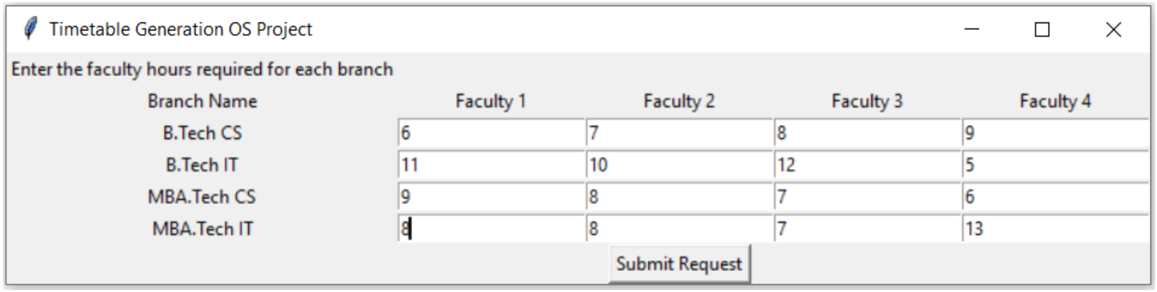
Quantum time splitter:


Start
lst[0, 0, 0, 0] i.e. initialising list with each element representing each branch j = 0
mylist[element] exists?
False
True
final_list=[]
j<4
False
True
False
element==j
True
increment lst[j] by 1and j++
while lst[0] != 0 or lst[1] != 0 or lst[2] != 0 or lst[3] != 0
True
j<4
False
True
False
lst[j]!=0
True
final_list.append(j) lst[j] --
Stop

Excel File Manipulation:


Start
faculty_list (from semaphore_algo), branch (dict for branch names), book (load excel file), sheet (sheet1 of excel file), r = 4, c = 2, i = 0
faculty_list[i] exists?
False
book (save excel file)
Stop
True
j = 0
i=i+1, r=r+1, c=2
False
faculty_list[i][j] exists?
True
k = 0
j=j+1
faculty_list[i][j][k] exists?
False
True
sheet.cell(row=r, column=c).value = branch[faculty_list[i][j][k]], c=c+1, k=k+1

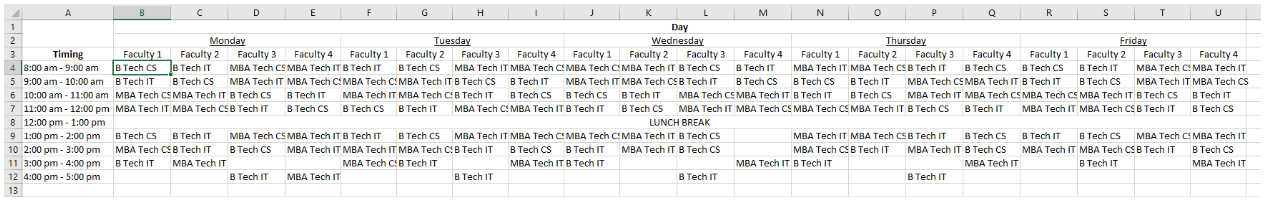
Input GUI :


Timetable Generation OS Project

Enter the faculty hours required for each branch

Branch Name	Faculty 1	Faculty 2	Faculty 3	Faculty 4
B.Tech CS	6	7	8	9
B.Tech IT	11	10	12	5
MBA.Tech CS	9	8	7	6
MBA.Tech IT	8	8	7	13

Output:


Day

	Monday				Tuesday				Wednesday			
Timing	Faculty 1	Faculty 2	Faculty 3	Faculty 4	Faculty 1	Faculty 2	Faculty 3	Faculty 4	Faculty 1	Faculty 2	Faculty 3	Faculty 4
8:00 am - 9:00 am	B Tech CS	B Tech IT	MBA Tech CS	MBA Tech IT	B Tech IT	B Tech CS	MBA Tech IT	MBA Tech CS	MBA Tech CS	MBA Tech IT	B Tech CS	B Tech IT
9:00 am - 10:00 am	B Tech IT	B Tech CS	MBA Tech IT	MBA Tech CS	MBA Tech CS	MBA Tech IT	B Tech CS	B Tech IT	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS
10:00 am - 11:00 am	MBA Tech CS	MBA Tech IT	B Tech CS	B Tech IT	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS	B Tech CS	B Tech IT	MBA Tech CS	MBA Tech IT
11:00 am - 12:00 pm	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS	B Tech CS	B Tech IT	MBA Tech CS	MBA Tech IT	B Tech IT	B Tech CS	MBA Tech IT	MBA Tech CS
12:00 pm - 1:00 pm										LUNCH BREAK		
1:00 pm - 2:00 pm	B Tech CS	B Tech IT	MBA Tech CS	MBA Tech IT	B Tech IT	B Tech CS	MBA Tech IT	MBA Tech CS	MBA Tech CS	MBA Tech IT	B Tech CS	
2:00 pm - 3:00 pm	MBA Tech CS	B Tech IT	B Tech CS	MBA Tech IT	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS	B Tech IT	MBA Tech IT	B Tech CS	
3:00 pm - 4:00 pm	B Tech IT	MBA Tech IT			MBA Tech CS	B Tech IT		MBA Tech IT	B Tech IT			MBA Tech IT
4:00 pm - 5:00 pm			B Tech IT	MBA Tech IT			B Tech IT				B Tech IT	

	Thursday				Friday			
Timing	Faculty 1	Faculty 2	Faculty 3	Faculty 4	Faculty 1	Faculty 2	Faculty 3	Faculty 4
8:00 am - 9:00 am	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS	B Tech CS	B Tech IT	MBA Tech CS	MBA Tech IT
9:00 am - 10:00 am	B Tech CS	B Tech IT	MBA Tech CS	MBA Tech IT	B Tech IT	B Tech CS	MBA Tech IT	MBA Tech CS
10:00 am - 11:00 am	B Tech IT	B Tech CS	MBA Tech IT	MBA Tech CS	MBA Tech CS	MBA Tech IT	B Tech CS	B Tech IT
11:00 am - 12:00 pm	MBA Tech CS	MBA Tech IT	B Tech CS	B Tech IT	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS
12:00 pm - 1:00 pm								
1:00 pm - 2:00 pm	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS	B Tech IT	B Tech CS	MBA Tech CS	MBA Tech IT
2:00 pm - 3:00 pm	MBA Tech CS	B Tech IT	MBA Tech IT	B Tech CS	MBA Tech IT	MBA Tech CS	B Tech IT	B Tech CS
3:00 pm - 4:00 pm	B Tech IT			MBA Tech IT		B Tech IT		MBA Tech IT
4:00 pm - 5:00 pm			B Tech IT					

Code:

GUI file:

```
from tkinter import *
from new_main import semaphore_algo

window = Tk()
window.title("Timetable Generation OS Project")


class ProvideException(object):
    def __init__(self, func):
        self._func = func

    def __call__(self, *args):
        try:
            return self._func(*args)
        except ValueError:
            text7 = Label(window, text="Please enter integer values only")
            text7.grid(row=7, column=0)
        except KeyboardInterrupt:
            text7 = Label(window, text="You hit a interrupt key like '
ctrl+c' or 'ctrl+v'. Please rerun the code.")
            text7.grid(row=7, column=0)
```

```
@ProvideException
def set_values():
    list_1 = [label3_1.get(), label3_2.get(), label3_3.get(),
label3_4.get()]       #Batch 1
    list_2 = [label4_1.get(), label4_2.get(), label4_3.get(),
label4_4.get()]       #Batch 2
    list_3 = [label5_1.get(), label5_2.get(), label5_3.get(),
label5_4.get()]       #Batch 3
    list_4 = [label6_1.get(), label6_2.get(), label6_3.get(),
label6_4.get()]       #Batch 4
    final_list = [list_1, list_2, list_3, list_4]
    print(list_1)
    print(list_2)
    print(list_3)
    print(list_4)
    print(final_list)

    fac_list_1 = []                 # Number of lectures by each batch
    fac_list_2 = []
    fac_list_3 = []
    fac_list_4 = []

    for faculty_no in range(0, 4):
        x = int(final_list[faculty_no][0])
        for hour_cnt in range(0, x):
            fac_list_1.append(faculty_no)

        x1 = int(final_list[faculty_no][1])
        for hour_cnt in range(0, x1):
            fac_list_2.append(faculty_no)

        x2 = int(final_list[faculty_no][2])
        for hour_cnt in range(0, x2):
            fac_list_3.append(faculty_no)

        x3 = int(final_list[faculty_no][3])
        for hour_cnt in range(0, x3):
            fac_list_4.append(faculty_no)

    print(fac_list_1)
    print(fac_list_2)
    print(fac_list_3)
    print(fac_list_4)

    semaphore_algo(fac_list_1, fac_list_2, fac_list_3, fac_list_4)


text1 = Label(window, text="Enter the faculty hours required for each
branch")
text1.grid(row=0)
```

```
text2 = Label(window, text="Branch Name")
text2_1 = Label(window, text="Faculty 1")
text2_2 = Label(window, text="Faculty 2")
text2_3 = Label(window, text="Faculty 3")
text2_4 = Label(window, text="Faculty 4")
text2.grid(row=1, column=0)
text2_1.grid(row=1, column=1)
text2_2.grid(row=1, column=2)
text2_3.grid(row=1, column=3)
text2_4.grid(row=1, column=4)

text3 = Label(window, text="B.Tech CS")
label3_1 = Entry(window)
label3_2 = Entry(window)
label3_3 = Entry(window)
label3_4 = Entry(window)
text3.grid(row=2, column=0)
label3_1.grid(row=2, column=1)
label3_2.grid(row=2, column=2)
label3_3.grid(row=2, column=3)
label3_4.grid(row=2, column=4)

text4 = Label(window, text="B.Tech IT")
label4_1 = Entry(window)
label4_2 = Entry(window)
label4_3 = Entry(window)
label4_4 = Entry(window)
text4.grid(row=3, column=0)
label4_1.grid(row=3, column=1)
label4_2.grid(row=3, column=2)
label4_3.grid(row=3, column=3)
label4_4.grid(row=3, column=4)

text5 = Label(window, text="MBA.Tech CS")
label5_1 = Entry(window)
label5_2 = Entry(window)
label5_3 = Entry(window)
label5_4 = Entry(window)
text5.grid(row=4, column=0)
label5_1.grid(row=4, column=1)
label5_2.grid(row=4, column=2)
label5_3.grid(row=4, column=3)
label5_4.grid(row=4, column=4)

text6 = Label(window, text="MBA.Tech IT")
label6_1 = Entry(window)
label6_2 = Entry(window)
label6_3 = Entry(window)
label6_4 = Entry(window)
text6.grid(row=5, column=0)
```

```
label6_1.grid(row=5, column=1)
label6_2.grid(row=5, column=2)
label6_3.grid(row=5, column=3)
label6_4.grid(row=5, column=4)

button1 = Button(window, text="Submit Request", command=set_values)
button1.grid(row=6, column=2)

window.mainloop()
```

Allocation using semaphores:

```
from quantum_time_splitter import quantum_splitter
from writing_excel import write

def semaphore_algo(f1,f2,f3,f4):
    fac_list=[]
    faculty_list = [[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -1],
[-1, -1, -1, -1], [-1, -1, -1, -1]]
                    ,[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -
1], [-1, -1, -1, -1], [-1, -1, -1, -1]]
                    ,[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -
1], [-1, -1, -1, -1], [-1, -1, -1, -1]]
                    ,[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -
1], [-1, -1, -1, -1], [-1, -1, -1, -1]]
                    ,[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -
1], [-1, -1, -1, -1], [-1, -1, -1, -1]]
                    ,[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -
1], [-1, -1, -1, -1], [-1, -1, -1, -1]]
                    ,[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -
1], [-1, -1, -1, -1], [-1, -1, -1, -1]]
                    ,[[-1, -1, -1, -1], [-1, -1, -1, -1], [-1, -1, -1, -
1], [-1, -1, -1, -1], [-1, -1, -1, -1]]]     # [Hour of Day][Day of
Week][Faculty]

    fac_list.append(quantum_splitter(f1))
    fac_list.append(quantum_splitter(f2))
    fac_list.append(quantum_splitter(f3))
    fac_list.append(quantum_splitter(f4))

    print('============================   REQUIREMENT HOURS
==================================')

    for i in range(4):
        print(fac_list[i])

    flag = 0

    print('==============================   ALLOCATED TIMETABLE
================================')

    for hour in range(0, 8):
        if flag == 1:
            break
        for day in range(0, 5):
            if len(fac_list[0]) == 0 and len(fac_list[1]) == 0 and 
len(fac_list[2]) == 0 and len(fac_list[3]) == 0:
                flag = 1
                break

            semaphore_list = []

            for i in range(4):
                if len(fac_list[i]) > 0:
                    try:
                        for x in range(0, 4):
                            if fac_list[i][x] not in semaphore_list:
                                faculty_list[hour][day][i] = 
fac_list[i][x]
                                semaphore_list.append(fac_list[i][x])
                                fac_list[i].pop(x)
                                break
                    except IndexError:
                        pass

    for hour in range(0, 8):
        print(faculty_list[hour])

    write(faculty_list)
```

Quantum Time Splitting:

```
def quantum_splitter(mylist):
    lst = [0, 0, 0, 0]

    for element in mylist:
        for i in range(4):
            if element == i:
                lst[i] += 1
                break
    final_list = []

    while lst[0] != 0 or lst[1] != 0 or lst[2] != 0 or lst[3] != 0:
        for i in range(4):
            if lst[i] != 0:
                final_list.append(i)
                lst[i] -= 1
    return final_list
```

Displaying output in Excel:

```
from openpyxl import load_workbook

def write(faculty_list):
    try:
        branch = {0: 'B Tech CS', 1: 'B Tech IT', 2: 'MBA Tech CS', 3:
'MBA Tech IT', -1: ''}

        book = load_workbook('Timetable.xlsx')
        sheet = book["Sheet1"]

        r, c = 4, 2
        for hour in faculty_list:
            for day in hour:
                for fac_no in day:
                    sheet.cell(row=r, column=c).value = branch[fac_no]
                    c = c+1
            if r == 7:
                r = r+2
            else:
                r = r+1
            c = 2

        book.save('Timetable.xlsx')
    except IOError:
        print("An Error occurred trying to write in the file.")
```

Limitation and Future scope

Though easily modifiable, the set input parameters can be looked at as a constraint or limitation.

For future there is a scope of adding functionalities like classroom and lab allotment, having some flexibility with respect to faculty and guest lectures instead of the fixed 8 hour duration etc.

List of References

BOOKS

Operating Systems: Internals and Design Principles - by William Stallings

LINKS

https://www.geeksforgeeks.org/python-writing-excel-file-using-openpyxl-module/

https://stackoverflow.com/questions/13381384/modify-an-existing-excel-file-using-openpyxl-in-python

https://www.lucidchart.com (for making flowcharts)

https://www.draw.io (for making flowcharts)