Department of Electrical and Computer Engineering

COEN 6311 – Software Engineering Fall 2022

MINI PROJECT REPORT

Submitted to

Dr. Tariq Daradkeh and Salah Harb

Submitted by

Srikanth Vadlamudi (ID:40232985)

Contents: Pg No.

1.Project description:

The project's goal is to create an Object oriented program for task assignment of a company with three hierarchy levels as follows 1.General Manager 2.Manager 3.Worker.

The company will have 10 members and two departments. General Manager has full access to assign tasks to managers and workers, can see the full status report with workers name, ID, assigned tasks, unassigned tasks and resolved tasks. General Manager will get notifications for newly assigned tasks.

Where as, department manager can only assign tasks to their department workers and need approval from general manager to cancel the department tasks. Managers will get notifications for newly assigned tasks similar to General Managers.

Workers can access all unassigned tasks of their Department and can assign task to themselves and they cannot cancel assigned tasks. The maximum number of tasks a worker can be assigned is three.

2.User stories:

1. As a General Manager, I want to assign tasks to managers and workers.

2. As a General Manager, I want to see the cancellation requests of department tasks by mangers.

3. As a General Manager, I want to able to approve cancellation of department tasks.

4. As a General Manager, I want to get notified for the newly assigned tasks for workers and departments.

5. As a General Manager, I want to see the status report.

6. As a Manager, I want to assign tasks to my department workers.

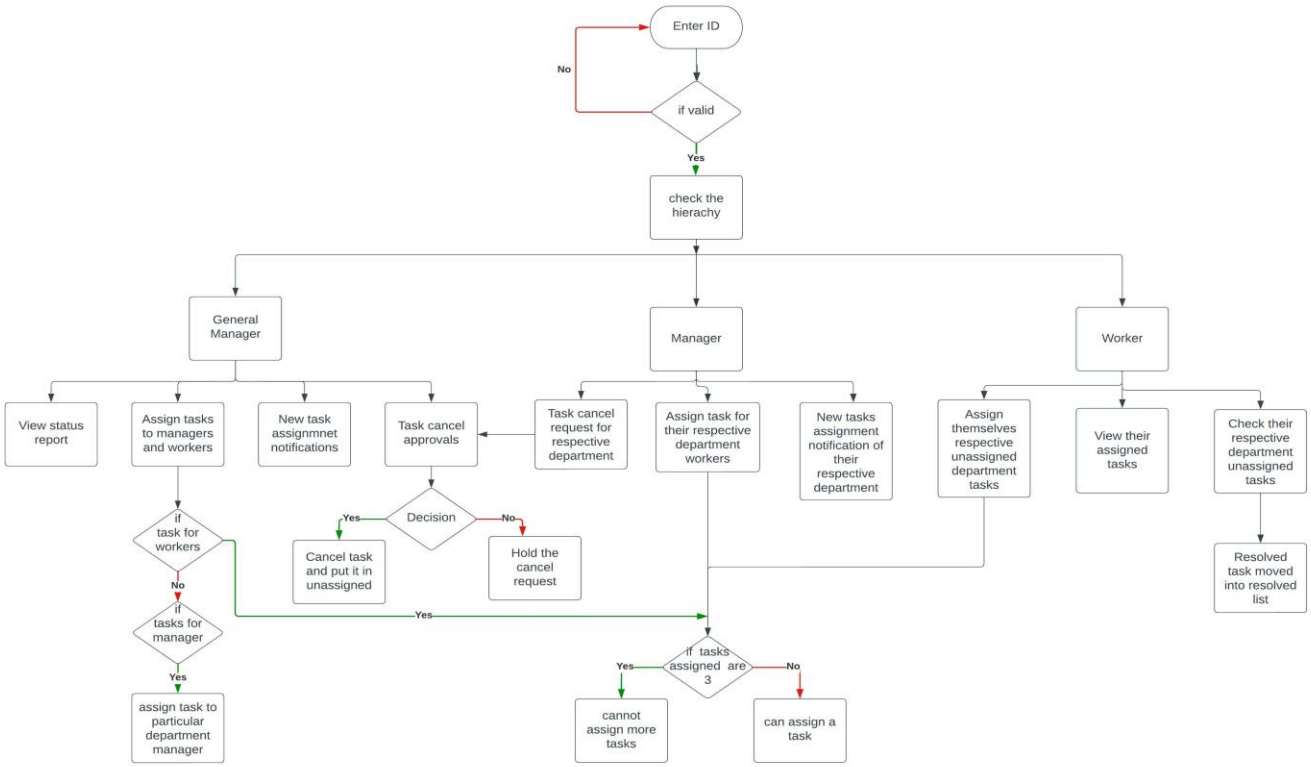
7. As a Manager, I want to send department task cancellation request to General Manager.

8. As a Manager, I want to get notifications for the newly assigned tasks for workers and department.

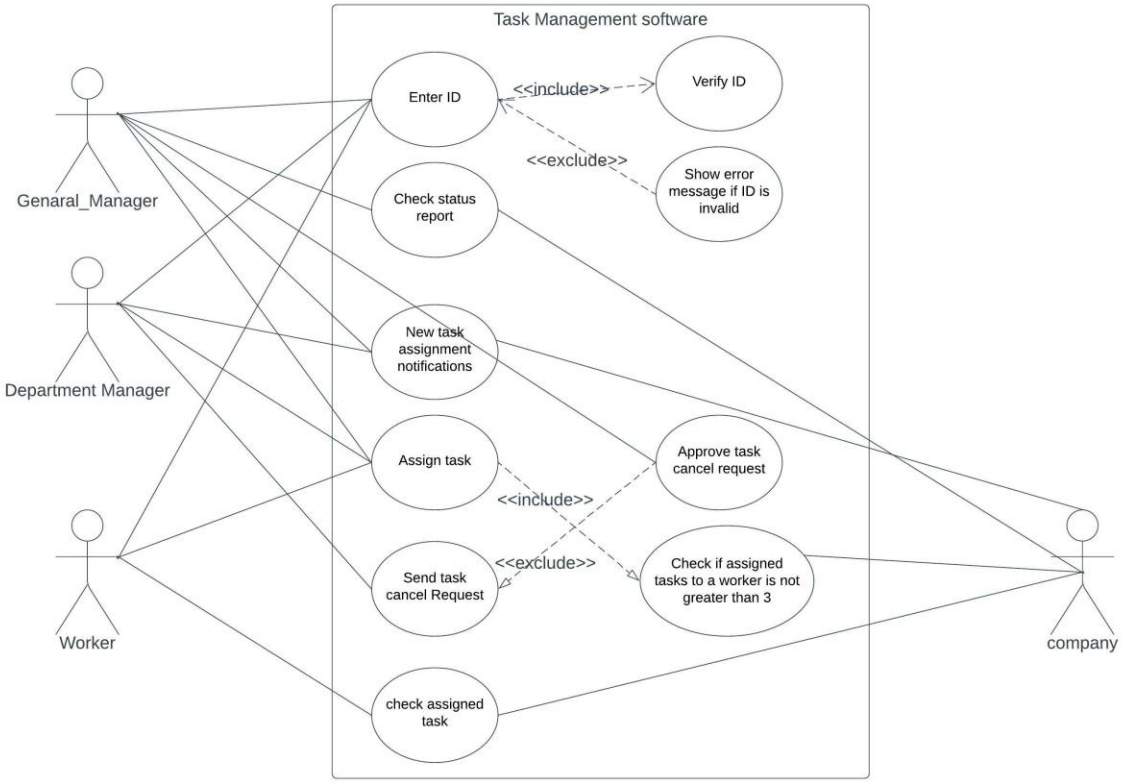
9. As a Worker, I want to check my assigned tasks.

10. As a Worker, I want to assign tasks to myself from my department tasks.

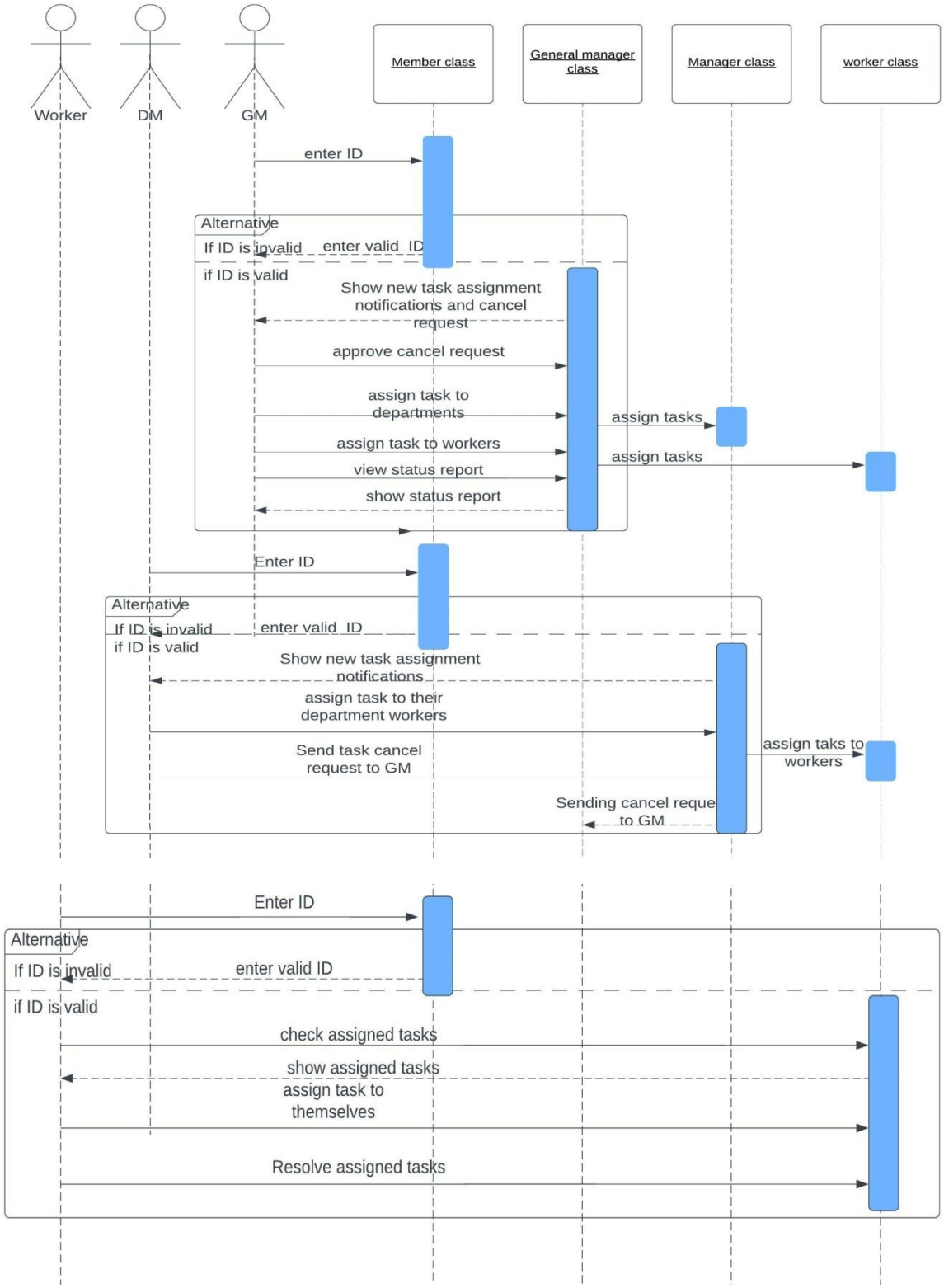
3. Business flow model:



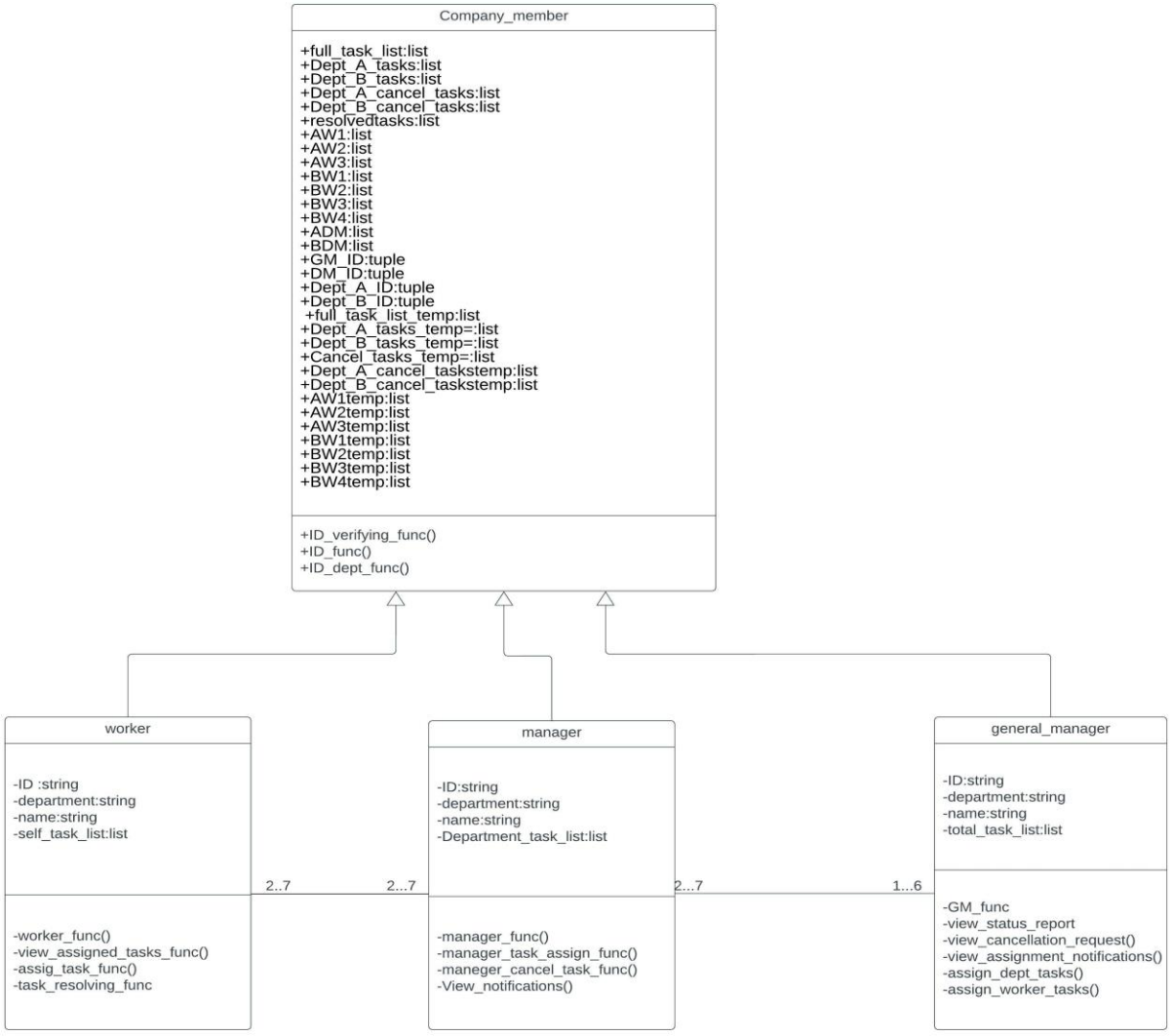
4. Use Case Diagram:



5. Sequence diagram:



6. Class diagram:

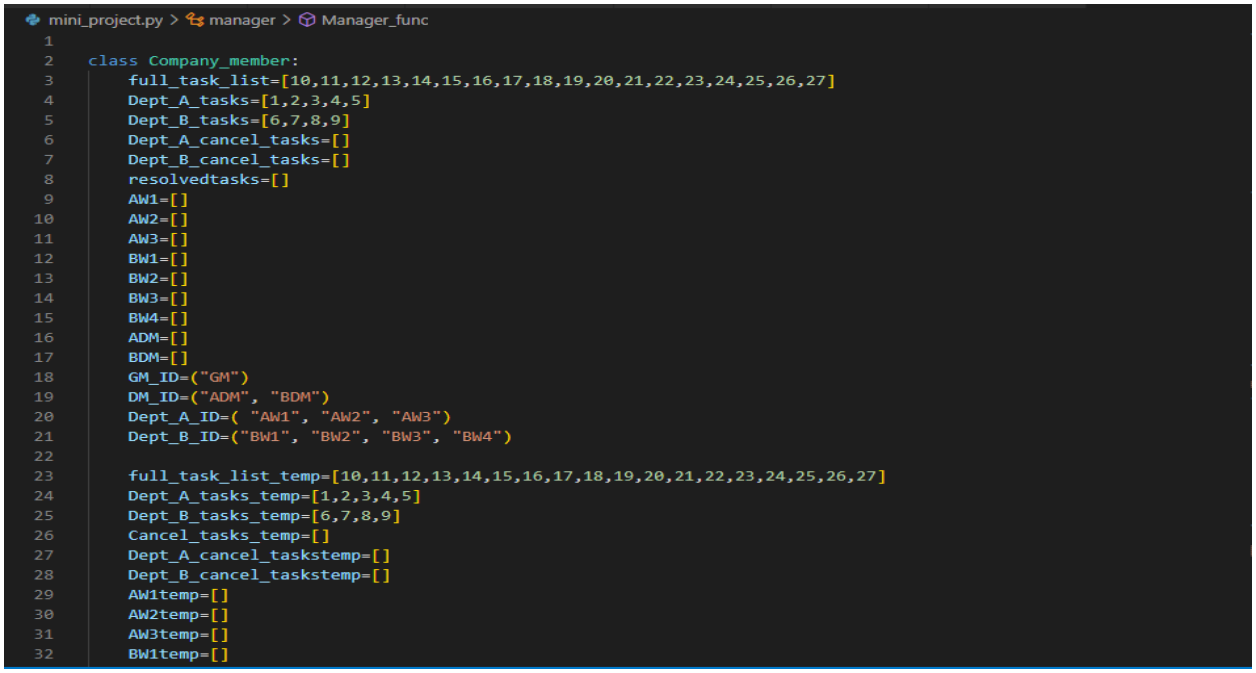


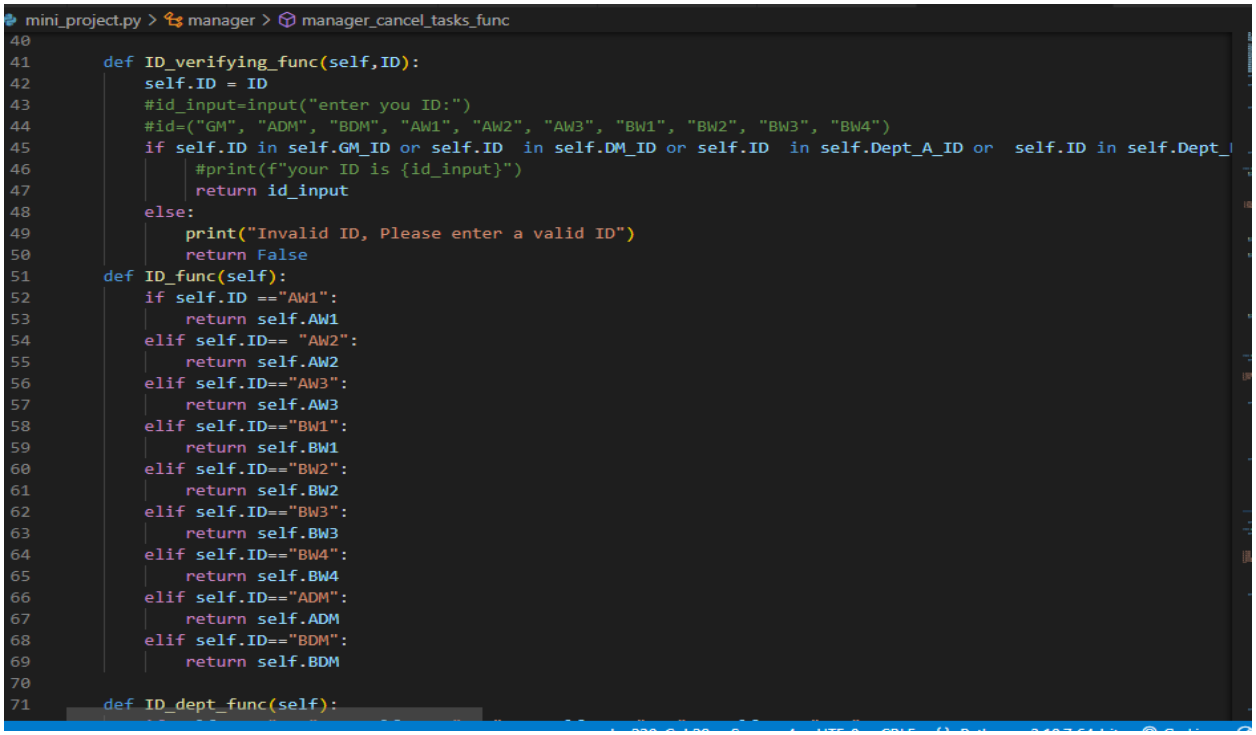
6.1 Code structure and class diagram relation:

Python language is used for this program and the use interface can be accessed via console menu.

From the class diagram, we can see that there are 4 classes which are company member class, worker class, manager class and general manager class. The worker, manager, general manager classes are inherited from the company member class. The inheritance let the 3 classes use the attributes defined in the company member class. The lists used to store the assigned task for each worker are defined as attributes and they needs to be accessed and modified by all the 3 classes, since manager and general manager can also assign tasks to workers as well as worker can assign tasks to him-self. So, the use of inheritance is justified.

When the program is executed, ID will be asked to navigate the program according to the hierarchy of the members. The ID will be verified. If the ID is invalid, the program will return to first step and asks for valid ID until a valid ID is entered. This functionality is executed by the ID_verifying_func() in the company member class. The ID_func() determines the member tasks list based on his ID and the ID_dept_func() fetches the department of the member from the ID.





If a valid ID is entered then the program processes the hierarchy of the ID and displays the functionalities of that particular member.

1. If the ID belongs to the General Manager, he has the following options and will be executed by particular methods in the class. The GM_func() calls the particular function of the general manager to be executed when the general manager chooses the option.

view_status_report()- To view status report.

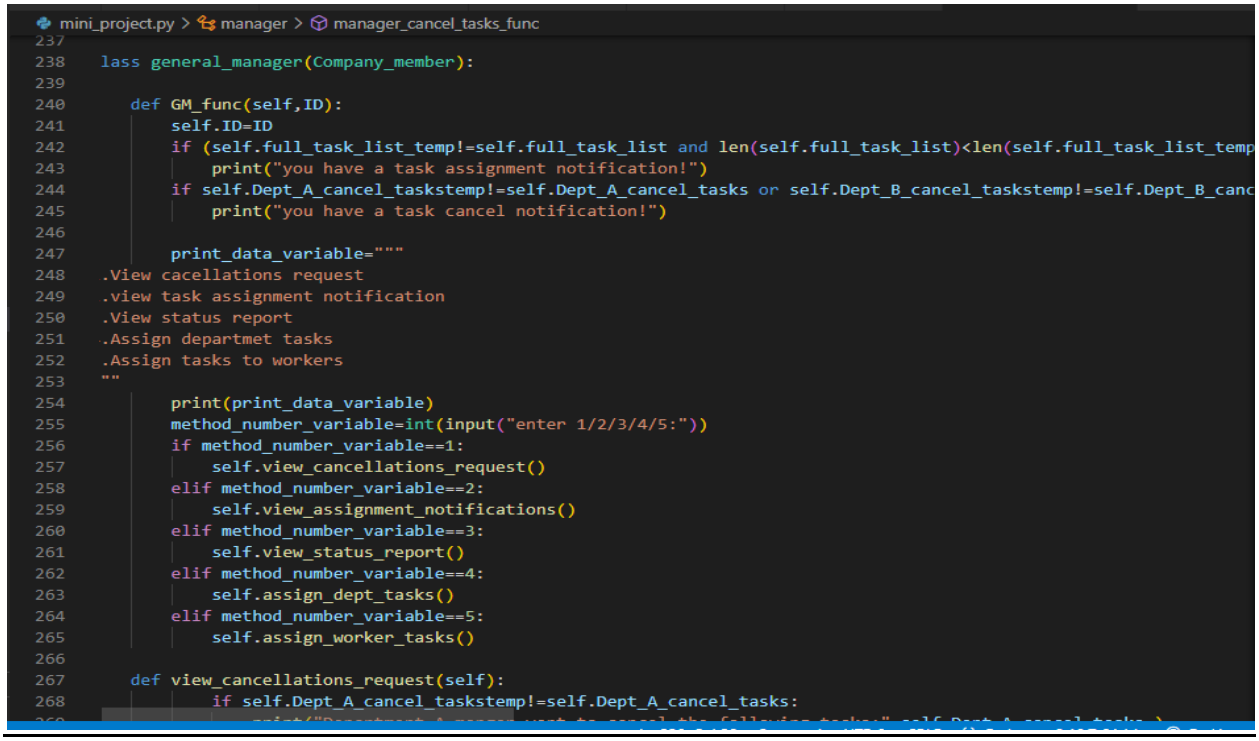
view_assigned_notifications()-To view newly assigned task notifications

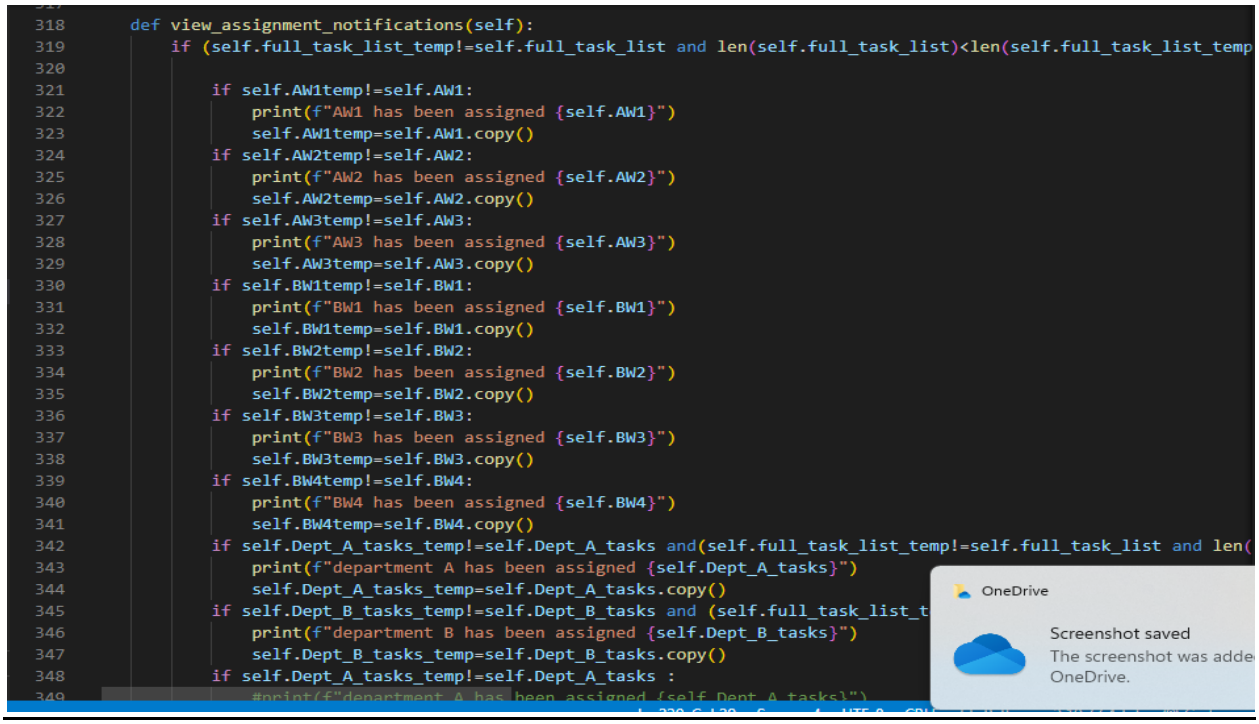
view_cancellation_request()-To view request for cancelling tasks from department manager

assign_dept_tasks()-To the assign task for departments

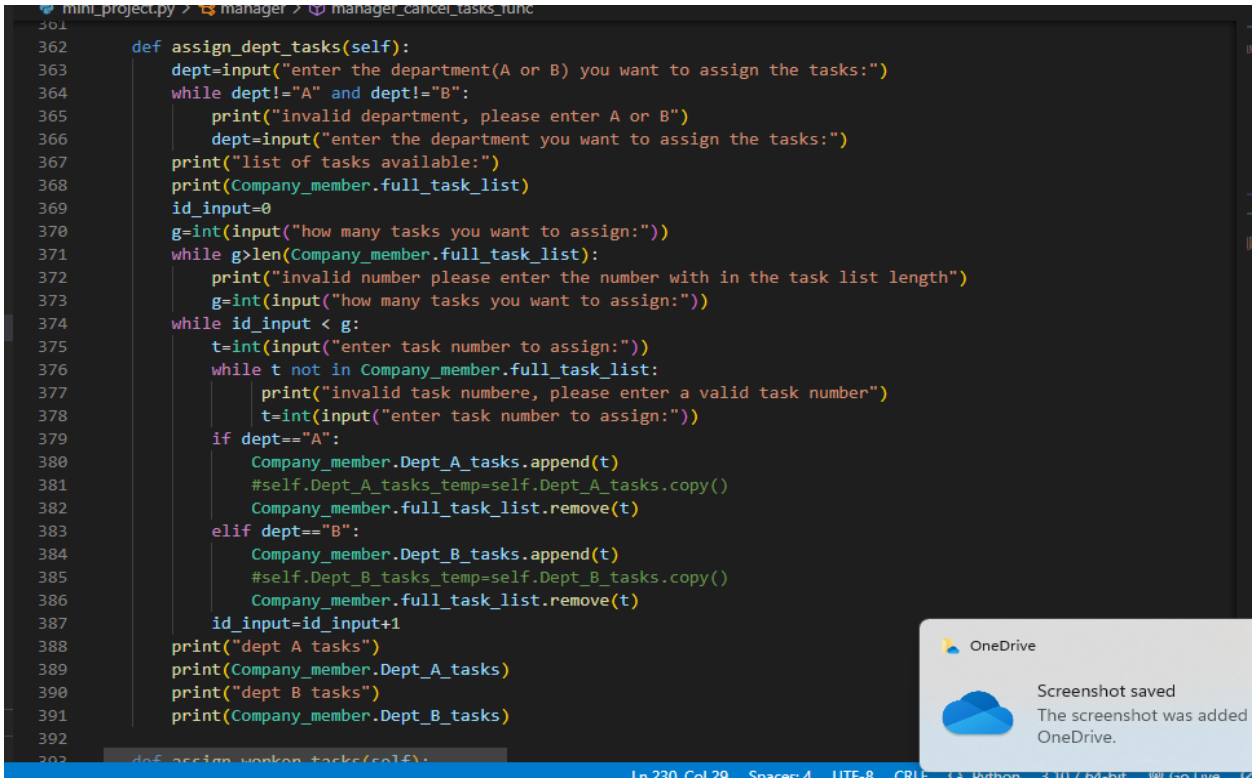
assign_worker_tasks()-To assign task for any worker .

If the General Manager decides to assign task to worker or, the assign_task_func() from the worker class is called and the process of assigning tasks for the worker or manager will be carried on. Since the method for assigned tasks will be same for worker or manager, the method which is defined in the worker class will be reused to avoid code duplication.





```
318     def view_assignment_notifications(self):
319         if (self.full_task_list_temp!=self.full_task_list and len(self.full_task_list)<len(self.full_task_list_temp
320
321             if self.AW1temp!=self.AW1:
322                 print(f"AW1 has been assigned {self.AW1}")
323                 self.AW1temp=self.AW1.copy()
324             if self.AW2temp!=self.AW2:
325                 print(f"AW2 has been assigned {self.AW2}")
326                 self.AW2temp=self.AW2.copy()
327             if self.AW3temp!=self.AW3:
328                 print(f"AW3 has been assigned {self.AW3}")
329                 self.AW3temp=self.AW3.copy()
330             if self.BW1temp!=self.BW1:
331                 print(f"BW1 has been assigned {self.BW1}")
332                 self.BW1temp=self.BW1.copy()
333             if self.BW2temp!=self.BW2:
334                 print(f"BW2 has been assigned {self.BW2}")
335                 self.BW2temp=self.BW2.copy()
336             if self.BW3temp!=self.BW3:
337                 print(f"BW3 has been assigned {self.BW3}")
338                 self.BW3temp=self.BW3.copy()
339             if self.BW4temp!=self.BW4:
340                 print(f"BW4 has been assigned {self.BW4}")
341                 self.BW4temp=self.BW4.copy()
342             if self.Dept_A_tasks_temp!=self.Dept_A_tasks and(self.full_task_list_temp!=self.full_task_list and len(
343                 print(f"department A has been assigned {self.Dept_A_tasks}")
344                 self.Dept_A_tasks_temp=self.Dept_A_tasks.copy()
345             if self.Dept_B_tasks_temp!=self.Dept_B_tasks and (self.full_task_list_t
346                 print(f"department B has been assigned {self.Dept_B_tasks}")
347                 self.Dept_B_tasks_temp=self.Dept_B_tasks.copy()
348             if self.Dept_A_tasks_temp!=self.Dept_A_tasks :
349                 #print(f"department A has been assigned {self.Dept_A_tasks}")
```

```
mini_project.py > manager > manager_cancel_tasks_func
361
362     def assign_dept_tasks(self):
363         dept=input("enter the department(A or B) you want to assign the tasks:")
364         while dept!="A" and dept!="B":
365             print("invalid department, please enter A or B")
366             dept=input("enter the department you want to assign the tasks:")
367         print("list of tasks available:")
368         print(Company_member.full_task_list)
369         id_input=0
370         g=int(input("how many tasks you want to assign:"))
371         while g>len(Company_member.full_task_list):
372             print("invalid number please enter the number with in the task list length")
373             g=int(input("how many tasks you want to assign:"))
374         while id_input < g:
375             t=int(input("enter task number to assign:"))
376             while t not in Company_member.full_task_list:
377                 print("invalid task numbere, please enter a valid task number")
378                 t=int(input("enter task number to assign:"))
379             if dept=="A":
380                 Company_member.Dept_A_tasks.append(t)
381                 #self.Dept_A_tasks_temp=self.Dept_A_tasks.copy()
382                 Company_member.full_task_list.remove(t)
383             elif dept=="B":
384                 Company_member.Dept_B_tasks.append(t)
385                 #self.Dept_B_tasks_temp=self.Dept_B_tasks.copy()
386                 Company_member.full_task_list.remove(t)
387             id_input=id_input+1
388         print("dept A tasks")
389         print(Company_member.Dept_A_tasks)
390         print("dept B tasks")
391         print(Company_member.Dept_B_tasks)
392
393     def assign_worken_tasks(self):
```

2. If the ID belongs to the Manager, he has the following options and will be executed by particular methods in the class. The manager_func() calls the particular function of the manager to be executed when the manager chooses the option.

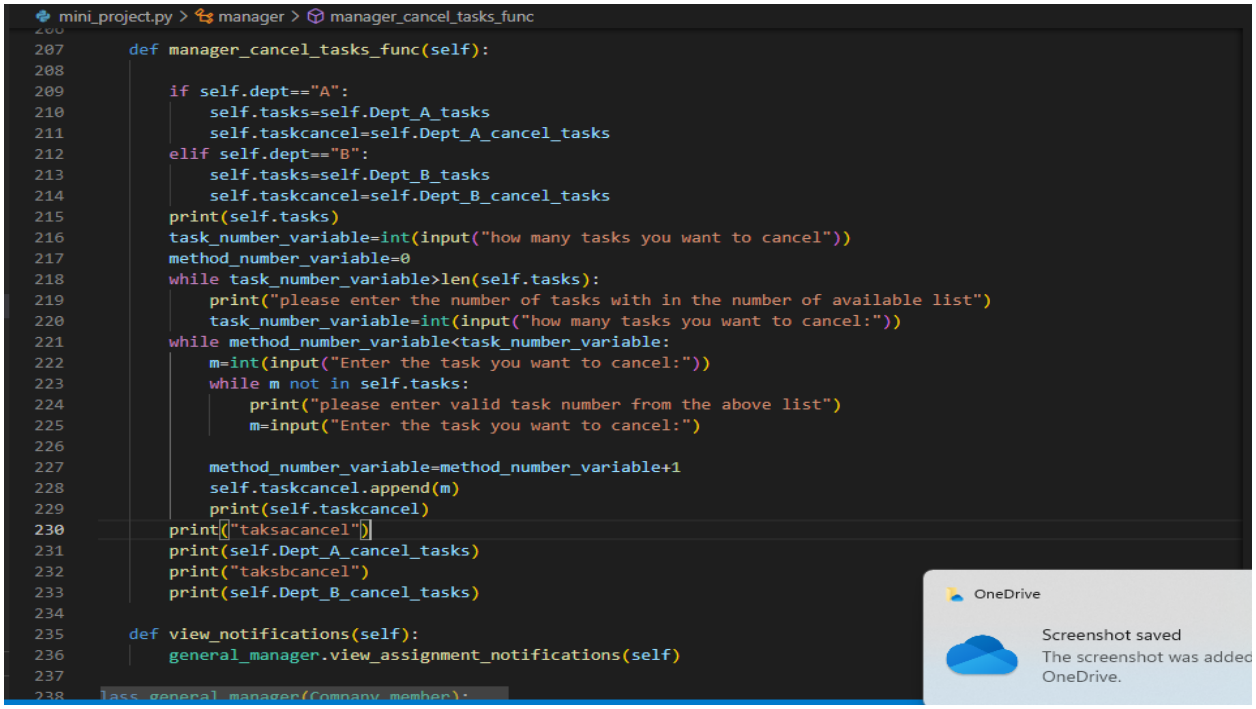
Manager_task_assign_func()- To assign tasks to workers of his department.

Manager_cancel_task_func()-To send approval request to general manager for cancelling department tasks.

Veiw_notifications(): To view the notifications of the newly assigned task for his department workers

If the Manager decides to assign task to worker or, the assign_task_func() from the worker class is called and the process of assigning tasks for the worker will be carried on. Since the method for assigned tasks will be same for worker or manager, the method which is defined in the worker class will be reused to avoid code duplication

To view notification for newly assigned tasks the method that is defined in the General Manager class is called and the view notification function is executed. Since the notification process is the same method from general manager is reused to avoid code duplication..



3. If the ID belongs to the worker, he has the following options and will be executed by particular methods in the class. The worker_func() calls the particular function of the worker to be executed when the worker chooses the option.

View_assigned_tasks()-To view which task are assigned to the worker.

Assign_task_func()-To assign task to himself from the unassigned department tasks.

Task_resolving_func()-To update the task that a worker resolved from his assigned tasks.

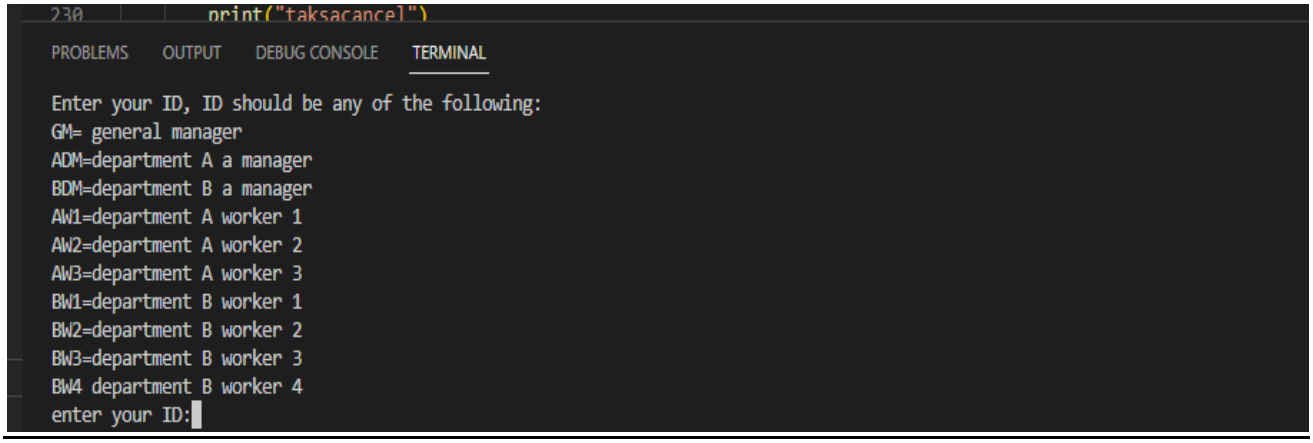
mini_project.py > manager > manager_cancel_tasks_func

```
79
80      @classmethod
81      def Worker_func(self,ID, ID_funcs, dept_variable, method_number):
82          self.ID=ID
83          self.ID_funcs=ID_funcs
84          self.dept_variable=dept_variable
85          self.method_number=method_number
86          if self.dept_variable=="A":
87              self.tasks=self.Dept_A_tasks
88          elif self.dept_variable=="B":
89              self.tasks=self.Dept_B_tasks
90
91          #print(f"  \nwelcome {self.ID}")
92          print_data_variable="""
93    1.check your task list
94    2.pick a task
95    3.complete a task
96          """
97          """
98          print(print_data_variable)
99
100         a=int(input("enter 1/2/3:"))
101         """
102         if self.method_number==1:
103             self.View_assigned_task_func()
104         elif self.method_number==2:
105             self.assign_task_func()
106         elif self.method_number==3:
107             self.task_resolving_func()
108
```



```
109     @classmethod
110     def View_assigned_task_func(self):
111         if len(self.ID_funcs)==0:
112             print("you don't have any tasks assigned, come back later!")
113         else:
114             print(self.ID_funcs)
115
116     @classmethod
117     def assign_task_func(self):
118
119         s=len(self.ID_funcs)
120         #print(s)
121         if s<0:
122             print("no tasks left to pick, come back later!")
123         else:
124             print('which task you would like to pick')
125             print(self.tasks)
126             t=int(input("enter the task number:"))
127
128             while t not in self.tasks:
129                 print("invalid task number,enter a valid task number from the above list")
130                 t=int(input("enter the task number:"))
131
132             if s<3:
133                 if t in self.tasks:
134                     self.ID_funcs.append(t)
135                     self.ID_funcs.sort()
136                     self.tasks.remove(t)
137             else:
138                 print(F"{self.ID} reached maximum task limit")
139
```

Output-User interface:



7.Plan of activities:

Activity	Estimated time	Actual time
Python tutorials	12 hours	18 hours
Writing code for worker functionalities	2 hours	2 hours
generalising code for all workers	4 hours	6 hours
Writing code for manager task cancel request	3 hours	3 hours
Writing code for general manager notification function	6 hours	8 hours
Writing code for genera manage task cancel approval	3 hours	4 hours
User stories	1 hour	1 hour
Business model flow diagram	1 hour	2 hours
Use case diagram	1 hour	2 hours
Sequence diagram	1 hour	3 hours
Class diagram	2 hours	4 hours