



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.....	3
2. User stories.....	3
3. Business model flow diagram.....	4
4. Use case diagram.....	5
5. Sequence diagram.....	6
6. Class diagram.....	7
6.1 Class diagram and code structure relation.....	7
7. Plan of activities.....	14

## **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.

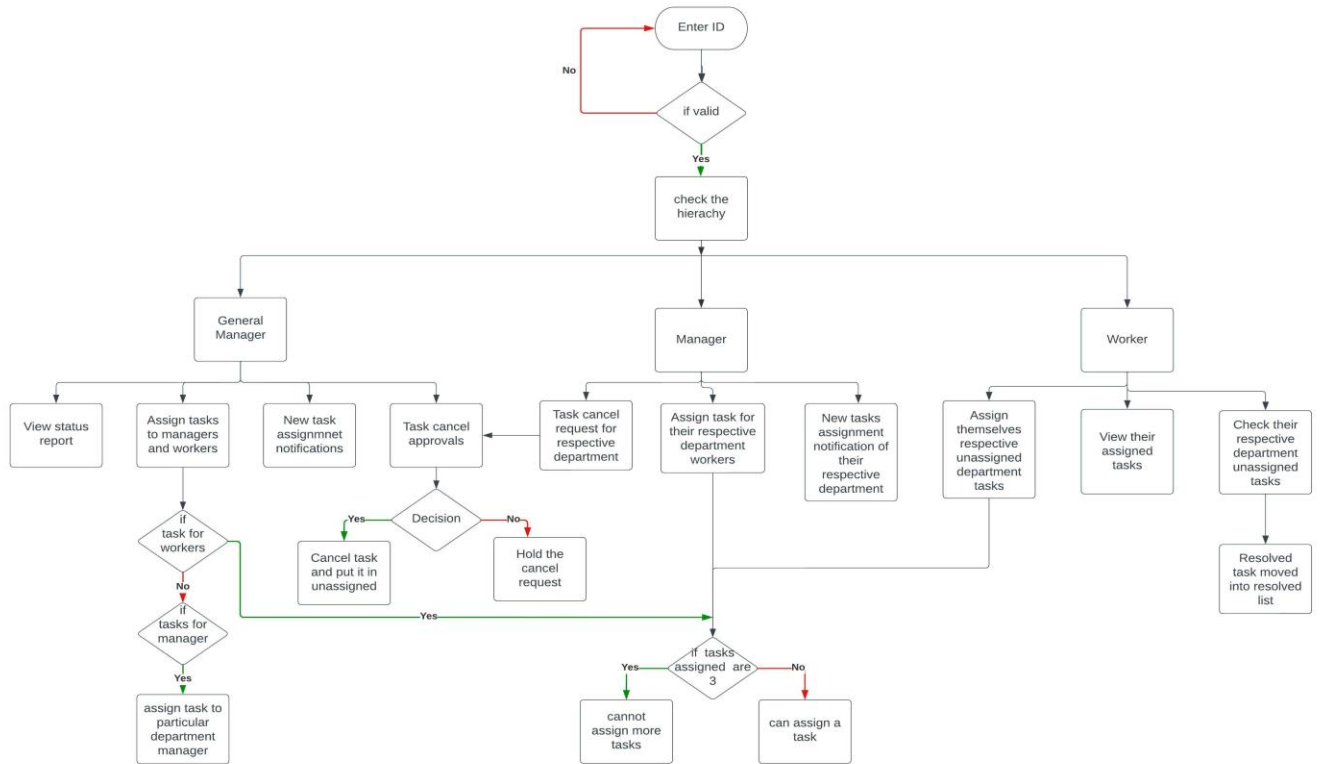
Where as, department manager can only assign tasks to their department workers and need approval from general manager to cancel the department tasks. Managers will get notifications for newly assigned tasks similar to General Managers.

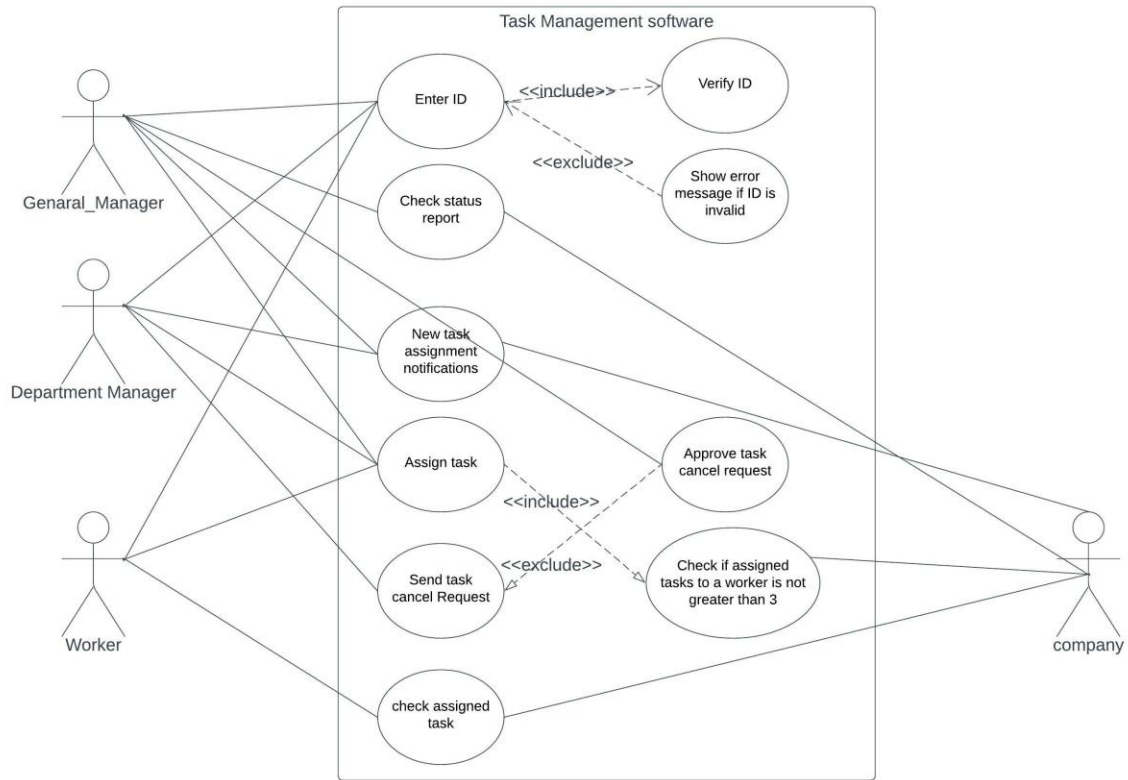
Workers can access all unassigned tasks of their Department and can assign task to themselves and they cannot cancel assigned tasks. The maximum number of tasks a worker can be assigned is three.

## **2.User stories:**

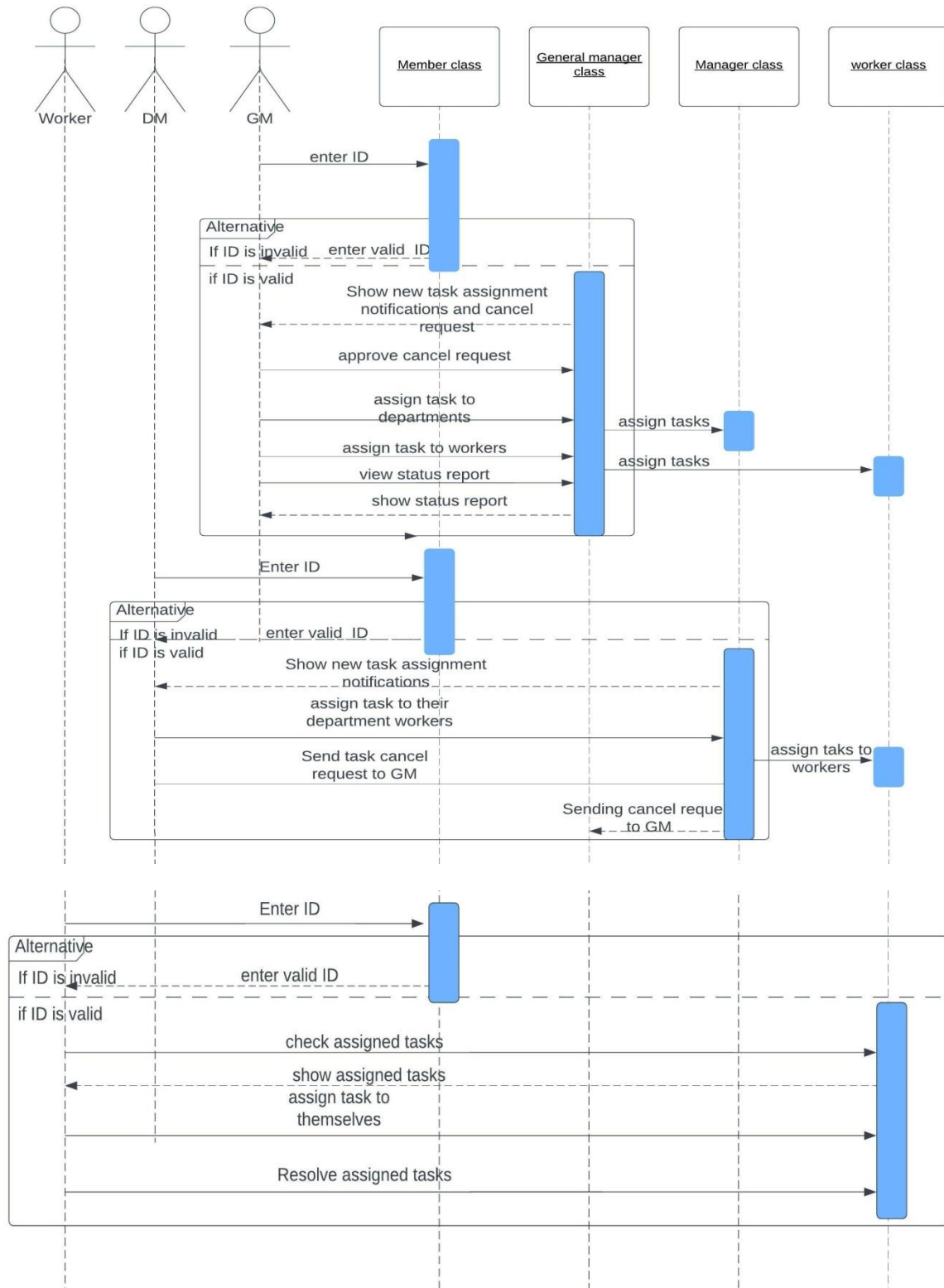
1. As a General Manager, I want to assign tasks to managers and workers.
2. As a General Manager, I want to see the cancellation requests of department tasks by managers.
3. As a General Manager, I want to able to approve cancellation of department tasks.
4. As a General Manager, I want to get notified 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:

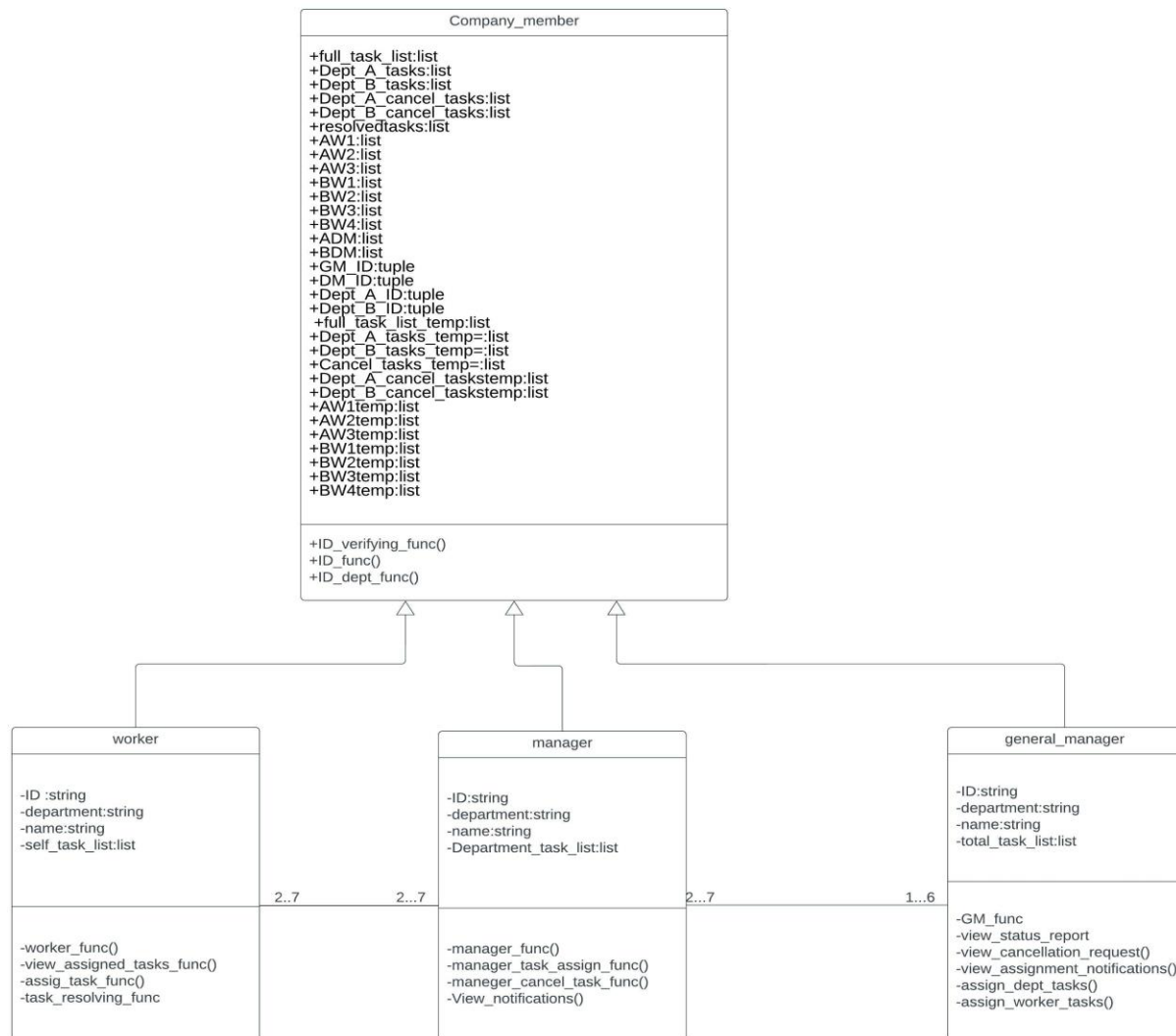




## 5. Sequence diagram:



## 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.

```

mini_project.py > manager > Manager_func
1
2 class Company_member:
3     full_task_list=[10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
4     Dept_A_tasks=[1,2,3,4,5]
5     Dept_B_tasks=[6,7,8,9]
6     Dept_A_cancel_tasks=[]
7     Dept_B_cancel_tasks=[]
8     resolvedtasks=[]
9     AW1=[]
10    AW2=[]
11    AW3=[]
12    BW1=[]
13    BW2=[]
14    BW3=[]
15    BW4=[]
16    ADM=[]
17    BDM=[]
18    GM_ID=("GM")
19    DM_ID=("ADM", "BDM")
20    Dept_A_ID= ("AW1", "AW2", "AW3")
21    Dept_B_ID= ("BW1", "BW2", "BW3", "BW4")
22
23    full_task_list_temp=[10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
24    Dept_A_tasks_temp=[1,2,3,4,5]
25    Dept_B_tasks_temp=[6,7,8,9]
26    Cancel_tasks_temp=[]
27    Dept_A_cancel_taskstemp=[]
28    Dept_B_cancel_taskstemp=[]
29    AW1temp=[]
30    AW2temp=[]
31    AW3temp=[]
32    BW1temp=[]

```

```

mini_project.py > manager > manager_cancel_tasks_func
40
41 def ID_verifying_func(self,ID):
42     self.ID = ID
43     #id_input=input("enter you ID:")
44     #id=("GM", "ADM", "BDM", "AW1", "AW2", "AW3", "BW1", "BW2", "BW3", "BW4")
45     if self.ID in self.GM_ID or self.ID in self.DM_ID or self.ID in self.Dept_A_ID or self.ID in self.Dept_B_ID:
46         #print(f"your ID is {id_input}")
47         return id_input
48     else:
49         print("Invalid ID, Please enter a valid ID")
50         return False
51 def ID_func(self):
52     if self.ID == "AW1":
53         return self.AW1
54     elif self.ID == "AW2":
55         return self.AW2
56     elif self.ID == "AW3":
57         return self.AW3
58     elif self.ID == "BW1":
59         return self.BW1
60     elif self.ID == "BW2":
61         return self.BW2
62     elif self.ID == "BW3":
63         return self.BW3
64     elif self.ID == "BW4":
65         return self.BW4
66     elif self.ID == "ADM":
67         return self.ADM
68     elif self.ID == "BDM":
69         return self.BDM
70
71 def ID_dept_func(self):

```



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.

```

mini_project.py > manager > manager_cancel_tasks_func
237
238 class general_manager(Company_member):
239
240     def GM_func(self,ID):
241         self.ID=ID
242         if (self.full_task_list_temp!=self.full_task_list and len(self.full_task_list)<len(self.full_task_list_temp):
243             print("you have a task assignment notification!")
244         if self.Dept_A_cancel_taskstemp!=self.Dept_A_cancel_tasks or self.Dept_B_cancel_taskstemp!=self.Dept_B_cancel_tasks:
245             print("you have a task cancel notification!")
246
247         print_data_variable=""
248         .View cancellations request
249         .view task assignment notification
250         .View status report
251         .Assign departmet tasks
252         .Assign tasks to workers
253     """
254         print(print_data_variable)
255         method_number_variable=int(input("enter 1/2/3/4/5:"))
256         if method_number_variable==1:
257             self.view_cancellations_request()
258         elif method_number_variable==2:
259             self.view_assignment_notifications()
260         elif method_number_variable==3:
261             self.view_status_report()
262         elif method_number_variable==4:
263             self.assign_dept_tasks()
264         elif method_number_variable==5:
265             self.assign_worker_tasks()
266
267     def view_cancellations_request(self):
268         if self.Dept_A_cancel_taskstemp!=self.Dept_A_cancel_tasks:
269             print("Department A manager want to cancel the following tasks:" self.Dept_A_cancel_tasks)

```

```

317
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_workon_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.

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

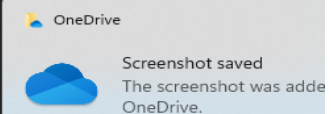
If the Manager decides to assign task to worker or, the `assign_task_func()` from the worker class is called and the process of assigning tasks for the worker will be carried on. Since the method for assigned tasks will be same for worker or manager, the method which is defined in the worker class will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

```

mini_project.py > manager > manager_cancel_tasks_func
143 @classmethod
144 def task_resolving_func(self):
145     if len(self.ID_funcs)==0:
146         print("you don't have any tasks assigned, come back later!")
147     else:
148         print(self.ID_funcs)
149         task_number_variable=int(input("which task you would like to complete:"))
150
151         while task_number_variable not in self.ID_funcs:
152             print("invalid task number,enter a valid task number from the above list")
153             task_number_variable=int(input("enter the task number:"))
154         self.ID_funcs.remove(task_number_variable)
155         self.resolvedtasks.append(task_number_variable)
156         print(f"{self.ID} new task list")
157         print(self.ID_funcs)
158         print("resolved tasks:")
159         print(self.resolvedtasks)
160
161 class manager(Company_member):
162
163     def Manager_func(self,ID, dept):
164         self.ID=ID
165         self.dept=dept
166         if (self.full_task_list_temp!=self.full_task_list and len(self.full_task_list)<len(self.full_task_list_temp):
167             print("you have a task assignment notification!")
168             print_data_variable=""
169             ..assign task to a worker
170             ..cancel task of the department
171             ..view notificaitons
172             """
173             print(print_data_variable)
174             method_number_variable=int(input("enter 1/2/3:"))

```



```

mini_project.py > manager > manager_cancel_tasks_func
207 def manager_cancel_tasks_func(self):
208
209     if self.dept=="A":
210         self.tasks=self.Dept_A_tasks
211         self.taskcancel=self.Dept_A_cancel_tasks
212     elif self.dept=="B":
213         self.tasks=self.Dept_B_tasks
214         self.taskcancel=self.Dept_B_cancel_tasks
215     print(self.tasks)
216     task_number_variable=int(input("how many tasks you want to cancel"))
217     method_number_variable=0
218     while task_number_variable>len(self.tasks):
219         print("please enter the number of tasks with in the number of available list")
220         task_number_variable=int(input("how many tasks you want to cancel:"))
221     while method_number_variable<task_number_variable:
222         m=int(input("Enter the task you want to cancel:"))
223         while m not in self.tasks:
224             print("please enter valid task number from the above list")
225             m=input("Enter the task you want to cancel:")
226
227         method_number_variable=method_number_variable+1
228         self.taskcancel.append(m)
229         print(self.taskcancel)
230     print(["tasksacancel"])
231     print(self.Dept_A_cancel_tasks)
232     print(["tasksbcancel"])
233     print(self.Dept_B_cancel_tasks)
234
235     def view_notifications(self):
236         general_manager.view_assignment_notifications(self)
237
238     class general_manager(company_member):

```

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:

```

230 | print("tasksancel")
    |
    | PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
    |
    | Enter your ID, ID should be any of the following:
    | GM= general manager
    | ADM=department A a manager
    | BDM=department B a manager
    | AW1=department A worker 1
    | AW2=department A worker 2
    | AW3=department A worker 3
    | BW1=department B worker 1
    | BW2=department B worker 2
    | BW3=department B worker 3
    | BW4 department B worker 4
    | enter your ID:
  
```

## 7.Plan of activities:

<u>Activity</u>	<u>Estimated time</u>	<u>Actual time</u>
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